AB INITIO MOLECULAR DYNAMICS (AIMD) A NEW APPROACH FOR DEVELOPMENT OF ACCURATE POTENTIALS

By

Milind M Malshe
Bachelor of Engineering
University of Mumbai
Mumbai, India
1998

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER of SCIENCE
December, 2004

AB INITIO MOLECULAR DYNAMICS (AIMD)
A NEW APPROACH FOR DEVELOPMENT OF ACCURATE POTENTIALS

Thesis Approved:

Dr. R. Komanduri
Thesis Advisor

Dr. S. Roy

Dr. H. Lu

Dr. M. Hagan

Dr. L. M. Raff

Dr. A. Gordon Emslie
Dean of the Graduate College

SUMMARY

In this study a new approach is presented for the development of accurate potential-energy hypersurfaces based on *ab initio* calculations that can be utilized to conduct molecular dynamics and Monte Carlo simulations to study chemical and mechanical properties at the atomistic level. The method integrates *ab initio* electronic structure calculations with the interpolation capability of multilayer neural networks. A sampling technique based on *novelty detection* is also developed to ensure that the neural network fitting for the potential energy spans the entire configuration space involved during the simulation. The procedure can be initiated using an empirical potential or direct dynamics simulation. The procedure is applied for developing the potential energy hypersurface for 5-atom clusters within a silicon workpiece. *Ab initio* calculations were performed using Gaussian 98 electronic structure program. Results for 5-atom silicon clusters representing the bulk and the surface structure are presented.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my parents and brother for their support, encouragement, love and confidence in me.

I would like to thank my advisor, Dr. R. Komanduri, for his inspiration, technical guidance, and financial support. I wish to express my sincere appreciation to Dr. L. M. Raff for introducing me to the concept of Molecular Dynamics and Monte Carlo simulations and for providing guidance and encouragement throughout the study. I would like to thank Dr. M. Hagan for introducing me to the field of Neural Networks and for his invaluable guidance and encouragement. I would like to thank Dr. P. M. Agrawal for his guidance in understanding various aspects of simulation techniques. This project is a result of coordinated effort linking simulation methods and neural networks. I would also like to thank Dr. S. Roy and Dr. H. Lu for their suggestions in this study. I'm grateful to Dr. R. Komanduri and my committee members for helping me to explore and develop new methods for improving simulation techniques.

This project is funded by grant DMI-0200327 from the National Science Foundation. I thank Dr. W. DeVries, Dr. G. Hazelrigg, Dr. J. Chen, and Dr. D. Durham of the Division of Design, Manufacturing, and Industrial Innovation, Dr. B. M. Kramer, Engineering Centers Division, and Dr. J. Larsen Basse, Tribology and Surface Engineering program, for their interest and support of this work. This project was also sponsored by a DEPSCoR grant on the Multiscale Modeling and Simulation of Material

Processing (F49620-03-1-0281). I thank Dr. Craig S. Hartley of the AFOSR for his interest in and support of this work.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

Computer simulation techniques play an important role in the study of various chemical, biological and mechanical processes at the atomistic level. By the comparing results of the simulations with experimental observations, theoretical model used in the simulation can be corrected and validated. Once validated, the simulations can be used to study the system under different conditions for which experimental results are not easily available. By analyzing the results from the simulations, experiments can then be performed under specific conditions to achieve desired results. Thus computer simulations can complement both theoretical and experimental approaches. In conjunction with the developments in computer technology, the use of simulations has greatly extended the range of problems that can be studied. In chemistry, simulations are performed to study reaction dynamics. In engineering, simulations are used to study the behavior of a material under varying conditions in various processes such as machining, indentation, and melting. Such simulations can provide a useful insight into important mechanisms such as phase transformation and dislocation dynamics, which otherwise is not easily understood using other techniques.

Simulation techniques can be broadly classified into two groups as, stochastic and deterministic. Stochastic simulations are based on statistical mechanics which relates

macroscopic properties such as pressure and temperature to the distribution and motion of atoms and molecules, whereas deterministic approach is based on classical mechanics. Examples of deterministic and stochastic approaches are molecular dynamics and Monte Carlo simulations respectively. Molecular dynamics simulations generate information at the microscopic level, such as atomic positions and velocities, by integrating the equations of motions and following the time evolution of a set of interacting atoms. Statistical mechanics deals with macroscopic systems from a molecular point of view, where macroscopic phenomenons are studied from the properties of individual molecules making up the system.

The most important part of MD/MC simulation is the development of the potential energy surfaces, which describe the interactions of atoms within a system. It should provide a realistic description of the interatomic interactions to match the experimental observations. The usual approach for developing a potential is to determine a functional form motivated by physical intuition and then to adjust the parameters either to *ab initio* data and/or some physical properties to come up with an empirical potential. Although, such empirical potentials provide a simple and physically interpretable description for the interatomic interactions their applicability is limited to the type of data to which it was fitted. Once fitted, there is no easy way to improve upon it, without refitting the data. A solution to this problem is to model the system using *ab initio* electronic structure calculations by solving Schrödinger's equation, to compute the potential energy and forces from the first principles. Such a technique can provide very accurate description of the interatomic interactions, but are computationally very extensive and hence limited only to small systems involving only a few atoms.

In this study a new approach is presented for the development of accurate potential-energy hypersurfaces based on *ab initio* calculations that can be utilized to conduct molecular dynamics and Monte Carlo simulations to study chemical, mechanical, and electrical properties at the atomistic level. The method integrates *ab initio* electronic structure calculations with the interpolation capability of multilayer neural networks. A sampling technique based on *novelty detection* is also developed to ensure that the neural network fitting for the potential energy spans the entire configuration space involved during the simulation.

Chapter 2 gives a literature review of various empirical potentials and different design approaches employed for modeling different situations. Chapter 3 explains the molecular dynamics technique, and various integration algorithms and implementation of periodic boundary conditions. Chapter 4 gives the classification and design of empirical potentials and details about some of the important empirical potentials developed for covalent materials, particularly silicon. Chapter 5 provides a detailed description of *ab initio* techniques. Different types of basis sets and techniques, such as Hartree-Fock method, electron correlation methods, such as configuration interaction, perturbation theory and density functional theory are discussed. Chapter 6 discusses multilayer neural networks, training algorithms and techniques to achieve an accurate fit. A sampling technique called *novelty detection* is also discussed. Chapter 7 presents the new technique for *ab initio* Molecular Dynamics, and discusses various steps involved. Chapter 8 shows the results for the silicon system. Chapter 9 summarizes the conclusions of this study.

CHAPTER 2

Literature Review

The most important part of the MD/MC simulation is the development of the potential energy surfaces, which can model the system under the consideration sufficiently close to the actual one. In recent years many empirical potentials for silicon such as Stillinger-Weber potential, Tersoff potential, Bolding-Anderson potential and Brenner potential have been developed and applied to number of different systems.

2.1 Literature review of design of interatomic potentials

Existing models for interatomic differ in the degree of sophistication, functional form, fitting strategy and the range of applicability where each one is designed for a specific problem. Not only surfaces and small clusters are difficult to model (Balamane *et al,* 1992; Kaxiras, 1996) even bulk material, including crystalline and amorphous phases, solid defects and liquid phase have not provided a transferable description by a single potential. The usual approach for developing an empirical potential is to arrive at a parameterized functional form motivated by physical intuition and then to find a set of parameters by fitting to either *ab initio* data or experimental observation, or both for various atomic structures. A covalent material presents a difficult challenge because complex quantum mechanical effects, such as chemical bond formation, hybrdization, metallization, charge transfer and bond bending must be described by an effective

interaction between atoms. In spite of a wide range of functional forms and fitting strategies most of these models have been successful in the regime for which they were parameterised, but have shown a lack of transferability. Balamane and Halicioglu (1992) performed a comparative study of six classical many-body potentials namely, Pearson, Takai, Halicioglu and Tiller potential; Biswas and Hamann potential; Stillinger and Weber potential; Dodson potential; Tersoff potential (T2); modified Tersoff potential (T3). They concluded that each has strengths and limitations, none appearing clearly superior to the others, and none being fully transferable.

The Stillinger-Weber potential (1985) was parameterized for experimental properties of solid and liquid silicon such as lattice energy and melting point. The model has a form of a third order cluster potential (Carlson, 1990) in which the potential energy is expressed as a linear combination of two and three-body terms, as:

$$E = \sum_{\substack{i,j \\ i<j}} v_2(i,j) + \sum_{\substack{i,j,k \\ i<j<k}} v_3(i,j,k), \qquad (2.1)$$

where $v_2(i,j)$ represents two-body interactions and $v_3(i,j,k)$ represents three-body interactions. The two-body interactions are expressed as :

$$v_2(r_{ij}) = \varepsilon\, f_2(r_{ij}/\alpha), \qquad (2.2)$$

$$\textit{and} \quad f_2 = A\left(B r^{-p} - r^{-q}\right) e^{(r-a)^{-1}},$$

where ε has energy units and α has length units.

The three-body interaction is expressed as a separable product of radial functions $g(r)$ and angular function $h(\theta)$. Thus:

$$v_3(i,j,k) = \sum_{ijk} g(r_{ij}) \cdot g(r_{ik}) \cdot \left(\cos(\theta_{ijk}) + \frac{1}{3}\right)^2, \qquad (2.3)$$

where θ_{ijk} is the bond angle formed by the bonds *ij* and *ik*, and $g(r)$ is a decaying function. The angular function has a minimum at $\theta = 109.5°$, i.e. the tetrahedral angle to represent the angular preference for sp^3 bonding. This potential gives a fairly realistic description of crystalline silicon, point defects, liquid and amorphous phases but provides inadequate description of under-coordinated or over-coordinated structures. Mistriotis *et al.* (1989) proposed an improvement over Stillinger-Weber potential which included four-body interactions as well. It was able to give good agreement with the melting point of the crystal and the geometries and the energies of the ground and low metastable states of silicon clusters.

Tersoff (1988; 1989) proposed an approach by coupling two-body and multi-body correlations into the model. The potential is based on bond order which takes into account local environment. The bond strength depends on the geometry- the more neighbors an atom has, the weaker the bond to each neighbor would be. The energy is the sum of repulsive and attractive interactions which depends on the local bonding environment. Thus:

$$E = \frac{1}{2}\sum_{\substack{i,j \\ i \neq j}} f_C(r_{ij}) \cdot \left[f_R(i,j) + b_{ij}(\zeta_{ij})\, f_A(i,j) \right], \qquad (2.4)$$

$$\text{and} \quad \zeta_{ij} = \sum_{\substack{i,j,k \\ j \neq i \\ k \neq i, k \neq j}} v_3(i,j,k).$$

The f_R and f_A terms are repulsive and attractive pair potentials, respectively and f_C is a cutoff function. The main feature of this form is the term b_{ij}. The idea is that the strength of each bond depends upon the local environment and is lowered when the number of neighbors is relatively high. The term ζ_{ij} defines the effective coordination number of

atom taking into account the relative distance of two neighbors $(r_{ij}-r_{ik})$ and the bond angle θ_{ijk}.

The Bolding-Anderson potential (1990) is a general from of Tersoff potential, in which the interaction between pair of atoms dependent on the environment. They suggested that clusters of four or more atoms have local minima on the potential energy surface corresponding to multiple configurations, which involve atoms with high coordination number and strained bonds angles. The increase in potential because of strained bond angles is offset by the large number of weak bonds formed. Their potential incorporated angle dependence that is different for clusters and crystals by introducing interference functions.

Bazant and Kaxiras (1997) developed environment dependent interatomic potential (EDIP) for bulk silicon. The interaction model included two-body, three-body terms which depend on the local atomic environment through an effective coordination number given by:

$$V_2(r,Z) = \Phi_R(r) + p(Z)\,\Phi_A(r),$$

where $\Phi_R(r)$ represents the short range repulsion of atoms, $\Phi_A(r)$ represents the attractive force of bond formation, and $p(Z)$ is the bond order, which determines the strength of the attraction as a function of the atomic environment, measured by the coordination Z. The explanation of the bond order is that, as an atom becomes overcoordinated beyond the ideal coordination beyond its valence, the bonds become more metallic, characterized by delocalized electrons (Carlsson, 1985; 1990; Abell, 1985; Pettifor, 1990). They provided a test of empirical theories of covalent bonding in solids using a procedure to invert *ab initio* cohesive energy curves. They showed that the inversion of *ab initio* cohesive

energy curves verified trends in chemical bonding across various bulk bonding arrangements consistent with theoretical predictions (Bazant and Kaxiras, 1996). Their results indicated that a coordination dependent pair interaction can provide a fair description of high symmetry crystal structures without requiring additional many-body interactions, and angular forces are needed only to stabilize the structure under symmetry breaking distortion, primarily for small coordinations.

In spite of various approaches, no potential has demonstrated a transferable description of molecule in all its forms leading to the conclusion that it may be too ambitious to attempt a simultaneous fit of all of the important atomic structures (bulk, crystalline, surface, amorphous, liquid phase and clusters), since qualitatively different aspects of bonding are at work in different types of structures.

Rahman and Raff (2001) presented analytical fitting to the *ab initio* data for dissociation dynamics of vinyl bromide. Their approach was to develop analytic functions for each of the energetically open reaction channels using functional forms suggested by physical and chemical considerations. The parameters of these functions are expressed as appropriate functions of system's geometry. These channel potentials are connected smoothly with parameterized switching functions that play a central role in fitting the potential barriers, the reaction coordinate curvatures, and the coordinate coupling terms obtained by the *ab initio* calculations.

Ischtwan and Collins (1994) have developed a moving interpolation technique in which the potential energy in the neighborhood of any point is approximated by Taylor's series using inverse bond length coordinates as the expansion variables. The data from *ab initio* calculations is used to obtain a set of initial Taylor series expansions. The

procedure expresses the potential at any configuration as a weighed average of the Taylor series about all points in the data set. Subsequently, the results are iteratively improved by computing trajectories on the Taylor series fitted surface and the internal coordinates are stored. These new points are added to the data set according to a weight factor that is determined by the relative density of points in the data set in the region of the new point. The criterion used to assess convergence of the iteratively improved potential to the final surface was computed using dynamic variable, such as reaction probability.

Jordan *et al.* (1995) used the moving interpolation trajectory sampling method to investigate the reaction dynamics.

Maisuradze *et al.* (2003) introduced an interpolating moving least squares (IMLS) method for the interpolation between the computed *ab initio* points. When the method is unrestricted, the least squares coefficients are obtained from the solution of a large matrix equation that must be solved repeatedly during a trajectory study.

Another approach involves bypassing the physical motivation behind the functional form of the potential in favor of elaborate fitting as described by Ercolessi and Adams, (1994). The potential involves combinations of cubic splines with effectively hundreds of adjustable parameters and the approach was force matching method, which involves fitting the potential to *ab initio* atomic forces of many atomic configurations including surfaces, clusters, liquids, and crystals. Potential form is defined by single variable functions- atomic coordinates, such as glue potential which is defined as:

$$V = \frac{1}{2}\sum_{i,j}\Phi(r_{ij}) + \sum_{i}U\left(\sum_{j}\rho(r_{ij})\right), \qquad (2.5)$$

where $\Phi(r_{ij})$ is a pair potential, $\rho(r_{ij})$ is a function of atomic density and *U(n)* is the glue function. They attempted to match the forces from first principle calculations for a large

set of configurations with those predicted by a classical potential, by minimizing the objective function, namely,

$$Z(\alpha) = Z_F(\alpha) + Z_C(\alpha), \qquad (2.6)$$

where Z_F contains *ab initio* data and Z_C consists of physical quantities from experimental results. If the emphasis is given to $Z_{F,}$ then the minimizing potential appears as the best approximation of the first principles system, with Z_C acting as a guide towards the correct region in the space of parameters. On the other hand, if the emphasis is on Z_C, then the method looks like conventional fit, but the Z_F term relieves the burden of guessing the functional form.

Another approach utilizes genetic algorithms and genetic programming. The idea is to let the computer program determine the functional form and come up with the best possible solution with minimal human input. The concept is based on genetics and evolution theory, in which different genetic representations of a set of variables are combined to produce a better solution. The technique is stochastic in nature rather than conventional gradient based approaches and hence the dimensionality of the problem does not pose a serious limitation. It works by randomly generating and exchanging functional elements (variables and arithmetic operators, such as addition and multiplication), and selecting the fittest individuals (which represent a set of parameters) assessed by comparing with target values, a population of potential evolves until a superior form emerges. Makarov and Metiu (1998) proposed a procedure by which genetic programming could be used to find the best functional form and the best set of parameters to fit the energy and derivatives from *ab initio* calculations. They suggested directed search which guides the computer to use a specified functional form without

limiting too much on the flexibility of the search. Directed search was used to fill in portions of human engineered functional template, without which the computer could not find a simple interpretable form even for a pair potential.

As an alternative to these methods Blank *et al.* (1993; 1995) explored an interpolation method based on feed forward neural networks. Functional approximation using neural networks does not require any assumption regarding the functional form of the potential. Multilayer feed forward neural networks could fit in principle to any real valued continuous function of any dimension to arbitrary accuracy provided it has sufficient number of neurons (Cybenko, 1989). Blank *et al.* used neural network to fit data sets produced by low dimensional models of CO molecule chemisorbed on a Ni(111) surface. The models were constrained versions of many dimensional empirical potential that had been used in the simulation of surface dynamics where it had been shown to give a reliable qualitative picture of the molecule- surface interaction.

Hobday *et al.* (1999) developed a neural network model for more complex C-N system for which the potential energy function was not parameterized. They used the neural network to fit the many-body term in the Tersoff functional form rather than fitting the entire potential. In contrast to the conventional network architecture, where a single input vector is presented to network, a set of N vectors where N is the number of neighbors of the two atoms i and j, not including i and j. The value of N depends on the i-j bond. For example if the i-j bond is a part of monocyclic chain, then $N=2$. If the i-j bond is tetrahedrally coordinated, then $N=6$. To present the local environment to the network, the bond order term is expressed as a function of i-j-k contributions, where k is the first neighbor of i or j. The input network consists of N vectors, where N is the number of

different *i-j-k* contributions. Each vector of the input data consisted of pairwise information, such as direction cosines, bond lengths, cutoff functions, and first neighbor information, such as its bond lengths and torsional angles and second neighbor information, such as the number of types of atoms used to describe the *i-j-k* atoms triplet. The size of the input vector was constant, but the number of input vectors *N* is a variable which could change during the simulation.

Car and Parrinello (1985) developed a unified approach for molecular dynamics and density functional theory. In this method, the ions are still moved classically but under the action of forces obtained by solving the electronic structure problem, thus eliminating the need for empirical potentials at the expense of much larger computational requirements. This technique is known as Car-Parrinello molecular dynamics method. The electron density in terms of occupied single particle orthonormal orbitals is written as:

$$n(r) = \sum_i \left|\Psi_i(r)\right|^2 . \qquad (2.7)$$

A point on the Born-Oppenheimer potential energy surface is given by the minimum with respect to Ψ_i of the energy functional. In the conventional formulation, minimization of the energy functional with respect to orbitals Ψ_i, subject to the orthonormality constraint leads to the self consistent Kohn-Sham equations. Its solution involves repeated matrix diagonalization. Car and Parrinello (1985) adopted a different approach by regarding the minimization of the Kohn-Sham (1965) functional as a complex optimization problem, which could be solved by applying optimization method called “simulated annealing” introduced by Kirkpatrick *et al.* (1983). In this approach, an objective function is minimized relative to the parameters with Boltzman type probability distribution via a

Monte Carlo procedure. In Car-Parrinello method the objective function is the total energy functional and the variational parameters are the coefficients of the expansion of Kohn-Sham orbitals. They found the simulated annealing strategy based on the MD, rather than on Metropolis Monte Carlo of Kirkpatrick *et al.* (1983) could be applied efficiently to minimize the Kohn-Sham functional. This technique called "dynamical simulated annealing" was found to be useful to study finite temperature properties.

2.2 Literature review of quantum mechanical calculations of silicon clusters

Raghavachari (1985; 1986; 1988) investigated the geometries and energies of silicon clusters using *ab initio* calculations. He considered several geometrical arrangements and electronic states. All geometries considered were optimized with several basis sets (minimal basis set- STO-3G, 6-31G) at Hartree-Fock level of theory and incase of open shell species (triplets, quintets, etc.) unrestricted Hartree-Fock was used. Then single point calculations were performed at these geometries using complete fourth order perturbation (MP4) theory with the polarized 6-31G* basis set. For the clusters of five silicon atoms, the geometries considered were- trigonal bipyramid, square pyramid, planar pentagon, linear structure, and the tetrahedral structure (which is the equilibrium structure for the bulk). He found the trigonal bipyramid with a singlet state to be the most stable structure of all the structures considered. In the case of square pyramid atom forms four equivalent bonds, all on one side of a plane which is not a stable structure and the molecule prefers to rearrange to more stable triangular bipyramid structure in spite of lower number of bonds present. They found the tetrahedral structure of five silicon atoms to have high energy. Though the central atom in such a cluster has

four bonds oriented in tetrahedral directions, the remaining four silicon atoms have only one bond each and are too distant from each other to interact favorably.

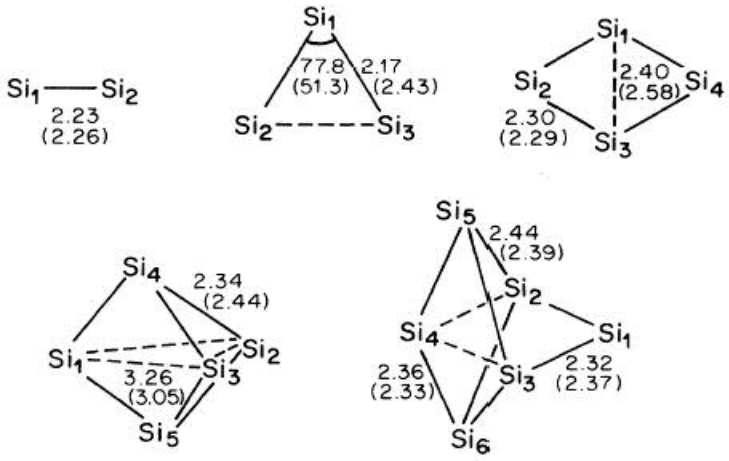


Fig 2.1 The geometries for the neutral and ionic clusters of Si_2 to Si_6, (Raghavachari, 1985). (Numbers in parentheses indicate bond lengths and bond angles for the ions).

Fig 2.1shows the ground state equilibrium structures for the neutral and ionic structures of Si2 to Si_6. Consideration of the structures Si_3 to Si_7 revealed that each cluster Si_n can be built from a smaller cluster Si_{n-1} by addition of a silicon atom at an appropriate edge or face capped bonding site. Edge capped structures were found to be favored in the case of smaller clusters while the face capped clusters became comparable in energy for the intermediate clusters. For example the rhombus structure of Si_4 could be considered as an edge capped triangular form, which is stable compared to the face capped tetrahedron.

Many atoms in larger clusters Si_7 to Si_{10} were found to have a coordination number of 6 (greater than 4) for the bulk. It indicated that unlike a close-packed metallic system there may be saturation of coordination at 6 and further bonding caused overcrowding and destabilization. He concluded that the principal differences between these clusters and the bulk were due to large fraction of surface atoms in the clusters.

These surface atoms provide a large driving force (analogous to the surface tension) to form more compact structures provided the resulting strain energy is not too high. In this sense there was a correspondence between the local coordination found in clusters and high pressure form of silicon *(β-tin* structure) which has hexacoordinate silicon in octahedral environment.

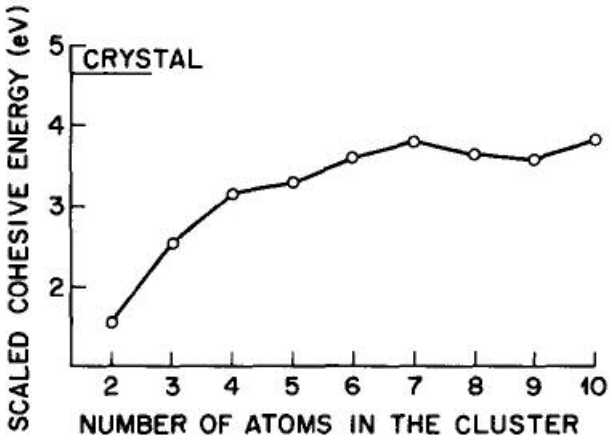


Fig 2.2 Scaled cohesive energy vs. cluster size for neutral silicon clusters calculated at the MP4/6-31G* level, (Raghavachari, 1985).

Fig 2.2shows the scaled cohesive energy as a function of the size of the cluster. The cohesive energy generally increases with some indication of special stability at cluster sizes 4, 7 and 10. Si_7 and Si_{10} approach the bulk cohesive energy more closely. Smaller clusters have low cohesive energy because of the difference in the number of bonds between the clusters and the bulk and not because of the nature of bonding.

The local relative stabilities of different size clusters could be compared by the incremental binding energy (which is defined as the energy required in the reaction $Si_n \rightarrow Si_{n-1}+Si$). Fig 2.3 shows an incremental binding energy as a function of the size of the cluster at HF/6-31G* and MP4/6-31G* levels.

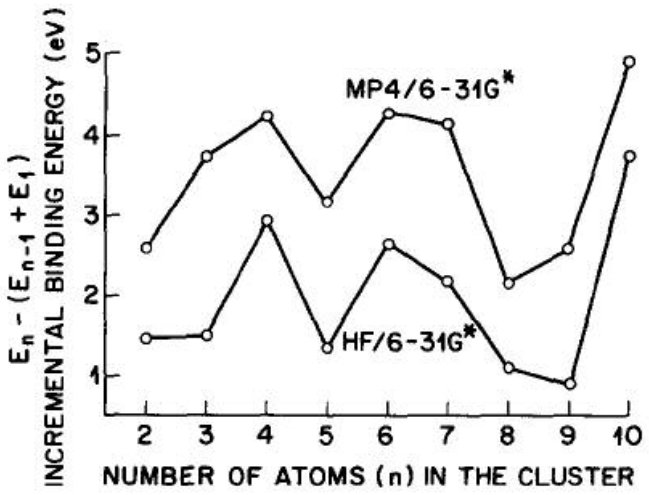


Fig 2.3 Incremental binding energy vs. cluster size for neutral silicon clusters calculated at the HF/6-31G* and MP4/6-31G* levels (Raghavachari and Rohlfing, 1988).

The peaks at cluster sizes of 4, 6, 7 and 10 indicate that these structures are locally stable, consistent with the prominent presence of 6, 7 and 10 in the mass spectral distributions of these clusters.

CHAPTER 3

Molecular Dynamics Simulation

Molecular Dynamics is a computer simulation technique where the time evolution of a set of interacting atoms is followed by integrating the equations of motions. While the continuum mechanics approach provides a better understanding of the processes at the macro and micro regime, it cannot be implemented to simulate the processes at the atomistic regime. Unlike in finite element analysis or other continuum approaches, where the nodes and distances are selected arbitrarily, in molecular dynamics it is based on more fundamental unit for a specific material such as: lattice spacing and inter-atomic distances. Thus the processes can be studied at the fundamental level. However, since a large number of atoms constitute any common material, one has to consider the interactions of several thousands of atoms even for nanometric study. Such simulations require large amount of computing resources but in return can provide an insight into processes happening at the smallest level, namely, atomistic level.

3.1 Brief history of molecular dynamics simulation

The molecular dynamics method was first introduced by Alder and Wainwright in the late 1950's (Alder and Wainwright, 1957, 1959) to study the interactions of hard spheres. The purpose of the paper was to investigate the phase diagram of a hard sphere system and in particular the solid and liquid regions. In a hard sphere system, particles

interact via instantaneous collisions, and travel as free particles between collisions. Gibson *et al.* (1960) studied the dynamics of radiation damage. It was the first example of molecular dynamics calculation with a continuous potential based on a finite time integration method. Rahman (1964) carried out perhaps the first simulation using a realistic potential for liquid argon. He studied properties of liquid argon using Lennard-Jones potential. Rahman and Stillinger (1974) carried out simulation of liquid water. Verlet calculated the phase diagram of argon using the Lennard-Jones potential, and computed correlation functions to test theories of the liquid state. The bookkeeping device, which became known as Verlet neighbor list was also introduced. The Verlet time integration algorithm was used. Phase transitions in the same system were investigated by Hansen and Verlet (1969). The number of simulation techniques has expanded since then; there exist now many specialized techniques for particular problems, including mixed quantum mechanical - classical simulations.

3.2 Applicability of MD simulations

Molecular dynamics follows the laws of classical mechanics. At the atomistic level, the system follows laws of quantum mechanics rather than classical mechanics. A classical description of the system involved can be used, when quantum effects, such as tunneling, and electronic excitations, play no essential role in the dynamics. In addition to this, the kinetic energy of a particle must be large enough, to ensure that the de Broglie wavelength is much smaller than the lattice constant of the solid. A simple test of the validity of the classical approximation is based on the de Broglie thermal wavelength defined as:

$$\Lambda = \sqrt{\frac{h^2}{2\pi \ M k_B T}}, \tag{3.1}$$

where M is the atomic mass and T is the temperature. The classical approximation is justified if $\Lambda << a$, where a is the mean nearest neighbor distance. The classical approximation is poor for lighter elements, such as hydrogen, helium. Moreover, quantum effects become important in any system when T is sufficiently low. The drop in the specific heat of crystals below the Debye temperature or the anomalous behavior of the thermal expansion coefficient, are well known examples of measurable quantum effects in solids. Hence, molecular dynamics results should be interpreted with caution in these regions.

An important issue of MD simulations is the temporal scale that can be explored. The maximum time step with which Newton's equations can be integrated numerically is inherently restricted to small values (about 1 fs), in order to reproduce atomic vibrations accurately. As a result molecular dynamics simulations require long computational times because the most interesting motions occurring during the simulation processes are very slow compared with the fast oscillations of bond lengths and bond angles that limit the integration time step. This limits the time scales that can be handled by MD in reasonable amount of actual clock time. MD simulations are usually limited to molecular processes happening in relatively short times, the so called "rare events" phenomenon in which infrequent processes are completed quickly, such as dissociation of a chemical bond which is a rapid process that is completed on the pico-second time scale. However the application of external load is longer by orders of magnitude compared with the dissociation time, making it difficult to observe such events at experimental speeds. In order to overcome the time scale limitation of MD, two approaches are usually followed.

In the first approach *coarse-graining*, the dynamic variables are divided into slow and fast degrees of freedom and averaging is performed on the fast degrees of freedom. In the second approach fast (and rare) trajectories between desired states are computed explicitly, and their probabilities (relative to non-reactive trajectories) are calculated.

3.3 Formulation of the differential equations of motion

The forces between the atoms are computed explicitly and the motion of the molecule is followed over a period of time using a suitable numerical integration method. Classical mechanics describes how physical objects move and how their positions change with time. Consider an isolated system consisting of N bodies with coordinates (x_i, y_i, z_i) where i=1, 2, 3… N (Goldstein, 1965). By isolated system it means that all other bodies are sufficiently remote to have a negligible influence on the system. Each of the N bodies is assumed to be small enough to be treated as a point particle. The position of the i^{th} body with respect to a given inertial frame is denoted as r_i (t). Its velocity and acceleration are given by v_i (t) = $\dot{r}_i(t)$ and a_i (t) = $\ddot{r}_i(t)$. Each atom has a mass m_i. From Newton's second law: $\vec{F}_i = m_i \vec{a}_i$

where $\vec{F}_i$ is the total force acting on an atom. This force is composed of a sum of forces due to each of the other interacting atoms in the system. If the force on i^{th} body due to j^{th} body is denoted as F_{ij}, then $\vec{F}_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} \vec{F}_{ij}$, where F_{ii} is zero, since there no force on i^{th} atom due to itself.

$$\vec{F}_i = m\frac{d^2\vec{r}_i}{dt^2} = \frac{d(m\vec{v}_i)}{dt} = \frac{d\vec{p}_i}{dt} \quad , \tag{3.2}$$

where $\vec{p}_i$ is the momentum of i^{th} body

The force on the atom *i* is the gradient of the potential energy function with respect to the coordinates of atom *i*.

$$\vec{F}_i = -\nabla_i V(\vec{r}_1, \vec{r}_2, ...\vec{r}_N), \tag{3.3}$$

where *V* is the potential energy function. $\vec{r}_i$ is the position vector of atom *i*, x_i, y_i, and z_i, are the Cartesian coordinates of atom *i*.

$$\vec{r}_i = x_i \hat{i} + y_i \hat{j} + z_i \hat{k},$$

and

$$\nabla_i = \frac{\partial}{\partial x_i}\hat{i} + \frac{\partial}{\partial y_i}\hat{j} + \frac{\partial}{\partial z_i}\hat{k} \ . \tag{3.4}$$

Now, force in the X- direction on atom *i* is given by:

$$F_x = \frac{d\ p_x}{dt} = \frac{d\,(m v_x)}{dt} = -\frac{\partial V}{\partial x} \ . \tag{3.5}$$

The velocity of atom *i* in X- direction is:

$$v_x = \frac{d\,x}{d\,t} = \frac{p_x}{m} . \tag{3.6}$$

Similar equations can be derived in Y- and Z- directions.

It should be noted that to update the position of an atom, one has to solve 6 coupled first order differential equations. If the number of atoms in the system is N, then the number of differential equations to be solved is 6N. If the potential function is a simple pairwise potential, such as Lennard- Jones or Morse potential, then the total number of terms in such a simple potential is N(N-1)/2. Thus, as an example, a system of 2000 atoms requires integration of 12,000 coupled first order differential equations of motion and about 2 million pairwise terms need to be calculated each time a derivative is evaluated. Such evaluations must be done for every integration step for every trajectory calculation. The computational time therefore, increases rapidly as the number of atoms in the system increase.

3.4 Time integration algorithms

The MD trajectories are defined by both position and velocity vectors and they describe the time evolution of the system. Accordingly the positions and velocities are propagated with a finite time interval by solving a set of coupled first order differential equations. Time integration algorithms are based on finite difference methods, where time is discretized on a finite grid. These are based on Taylor series expansion truncated after some finite number of terms. Truncation error varies as $(\Delta t)^n$, where Δt is the time step and *n* is the order of the method (which depends on the number of time derivatives considered in the expansion). Hence, for higher order methods, the truncation error drops off rapidly even with a small reduction in the time step. The idea of the numerical integration of Newton's equations of motion is to find an expression that defines positions at time $t + \Delta t$ in terms of already known positions and their time derivatives, such as velocities and accelerations at time *t*. Thus:

$$
\begin{aligned}
r(t+\Delta t) &= r(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2 + \ldots \\
v(t+\Delta t) &= v(t) + a(t)\Delta t + \frac{1}{2}b(t)\Delta t^2 + \ldots \\
a(t+\Delta t) &= a(t) + b(t)\Delta t + \ldots
\end{aligned}
\tag{3.7}
$$

3.4.1 Verlet algorithm

This is perhaps the most widely used method of integrating the equations of motion, initially developed by Verlet (1967) and attributed to Stömer (Gear, 1971). The Verlet algorithm uses acceleration at time *t* and positions at time *t* and $t-\Delta t$ to calculate the new positions at time $t+\Delta t$. The algorithm is time reversible, and given the conservative force field, it guarantees the conservation of the linear momentum.

$$r(t+\Delta t) = r(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2 + \ldots \quad \text{(3.8 A)}$$

$$r(t-\Delta t) = r(t) - v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2 + \ldots \quad \text{(3.8 B)}$$

Adding Eqn. (3.8 A) and (3.8 B), and neglecting higher order terms,

$$r(t+\Delta t) = 2r(t) - r(t-\Delta t) + a(t)\Delta t^2$$

and the velocity is computed as,

$$v(t) = \frac{r(t+\Delta t) - r(t-\Delta t)}{2\Delta t}$$

The basic Verlet algorithm does not uses velocities explicitly which may introduce numerical imprecision, since the local error in updating the positions is proportional to Δt^4, whereas the error in updating the velocities is proportional to Δt^2.

3.4.2 Leap- frog Verlet algorithm

This is a modification of the Verlet algorithm in which velocities are computed at half time step as well. The algorithm is given as,

$$r(t+\Delta t) = r(t) + \Delta t\, v(t+\frac{1}{2}\Delta t) \quad \text{(3.9)}$$

$$v(t+\frac{1}{2}\Delta t) = v(t-\frac{1}{2}\Delta t) + a(t)\Delta t$$

The advantage of this method is that the velocities are explicitly calculated, but the disadvantage is that they are not calculated at the same time as the positions. The velocities at time *t* can be approximated as,

$$v(t) = \frac{1}{2}\left(v(t-\frac{1}{2}\Delta t) + v(t+\frac{1}{2}\Delta t) \right) \quad \text{(3.10)}$$

3.4.3 Velocity Verlet algorithm

The velocity Verlet algorithm provides a better description of the velocities, without the need to compute the velocities at half the time step.

$$r(t+\Delta t) = r(t) + v(t)\Delta t + \frac{1}{2}a(t)\,\Delta t^2 \qquad (3.11)$$
$$v(t+\Delta t) = v(t) + \frac{1}{2}\left[a(t) + a(t+\Delta t)\right]\Delta t$$

3.4.4 Beeman algorithm

Beeman algorithm (1976) provides more accurate expression for velocities

$$r(t+\Delta t) = r(t) + v(t)\Delta t + \frac{2}{3}a(t)\Delta t^2 - \frac{1}{6}a(t-\Delta t)\Delta t^2 \qquad (3.12)$$
$$v(t+\Delta t) = v(t) + \frac{1}{3}a(t+\Delta t)\Delta t + \frac{5}{6}a(t)\Delta t - \frac{1}{6}a(t-\Delta t)\Delta t$$

It should be noted that these algorithms assume that the force remains constant while updating the positions from t to $t+\Delta t$. To circumvent this assumption, higher order terms involving higher time derivatives should be included in the algorithm.

It may be noted that the above mentioned methods are self starting methods i.e. it is only necessary to know the positions and velocities at time $t=t_0$. An estimate of accuracy during the integration can be obtained by monitoring the total energy of the system and the angular momentum which should be conserved. Step size reduction and back integration are other resources to check the accuracy of the integration results. Predictor-corrector methods provide an automatic error estimate at each integration step, allowing the program to employ a variable step size to achieve a specified accuracy. However, these methods are not self starting. Therefore it is necessary to use velocity Verlet or similar single step methods to start the integration.

3.5 Methods to improve computational speed

3.5.1 Neighbor list and book-keeping techniques

In the MD/MC simulations distances of atom *i* to all other atoms in the system are computed. For the calculation of potential and forces only the atoms that are separated by distances smaller than the potential cutoff are considered. The time to calculate all pair bond distances is proportional to N^2, where N is the number of atoms in the system. Verlet (1967) suggested a technique for improving the speed by maintaining a list of neighbors of a particular atom, which is updated at intervals. Between the updates of the neighbor list, only the bond distances of atoms appearing in the list are computed. This procedure reduces the computational time to some extent.

3.5.2 Cell structures and link lists

As the size of the system increases the conventional neighbor list becomes too large to store easily and testing of every pair in the system is inefficient. An alternative method of keeping track of neighbors for large systems is the cell index method. (Quentrec and Brot, 1975; Hockney and Eastwood, 1981). The cubic simulation box is divided into a regular lattice of M x M x M cells. A 2- dimensional representation is shown in Fig 3.1.These cells are chosen such that the side of the cell $l=L/M$ is greater than the cutoff distance of the potential.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Fig 3.1 Cell index method

The first part of the method involves sorting all atoms into appropriate cells. For this 2-dimensional example, neighbors of any atom in the cell 13 can be found in cells 13, 7, 8, 9, 12, 14, 17, 18, 19. Now searching for the neighbors is fast and efficient. For a 2-dimensional system there are approximately $N_c = N/M^2$ atoms in each cell and for a 3-dimensional system there would be $N_c = N/M^3$ atoms in each cell. Using the cell structure method only $9NN_c$ pairs have to computed for a 2-dimensional system and in 3-dimensional system $27NN_c$ pairs as opposed to $½\ N(N-1)$ in the conventional way. The cell structure can be set up and implemented in a linked list. The linked list array stores the number of the next atom in the cell, i.e. the list array element for an atom is the next of next atom in the cell and so on.

3.6 Periodic boundary conditions (PBC)

No matter how large the simulated system is its number of atoms would be negligible compared to the number of atoms that make up the macroscopic specimen (of the order of 10^{23}). As a result, in the simulation model the ratio of the number of surface atoms to the total number of atoms would be much larger than in reality, causing surface effects to be much more important than what they would be. A solution to this problem is the use of periodic boundary conditions. Periodic boundary is a mathematical formulation

to make a simulation that contains only a few hundred atoms to behave as though it is infinite in size. In doing so it also removes the surface effects since atoms near the boundary have less neighbor than the atoms inside. While using PBCs, the atoms are considered to be enclosed in a box, and this box is replicated by translation in all three Cartesian directions to form an infinite lattice. In effect every atom in the simulation box has an exact duplicate in each of the surrounding cells with the same velocities.

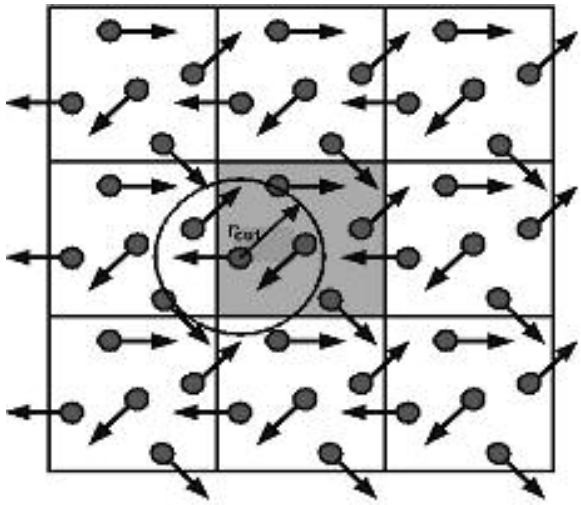


Fig 3.2 Representation of periodic boundary condition (PBC) and the simulation cell

Fig 3.2 shows the representation of the periodic boundary along with the main simulation cell. r_{cut} is the cutoff used in the potential.

If an atom is located at a position r in the box then this particle represents an infinite set of particle located at:

$$r + la + mb + nc, \tag{3.13}$$

where l, m and n are integer numbers, and a, b and c indicates the vectors corresponding to the edges of the box. All these image particles move together, but only one of them is represented in the computer program. Each particle should be thought of as interacting not only with other particles in the box but also with their images in the surrounding boxes. Whenever an atom leaves the simulation cell, it is replaced by another with the

same velocity entering from the opposite cell face, so the number of atoms in the simulation cell is conserved. Therefore no atom feels any surface effects.

An artifact of this technique is that if the simulation cell is too small and the cutoff radius is greater than half the length of the simulation cell then an atom *i* can interact with atom *j* as well as its image j_1 in the neighboring cell. As a result the interaction between atoms *i* and *j* is effectively counted twice as: *i-j* and $i\text{-}j_1$. The minimum image criterion states that if the box size is greater than 2 r_c, where r_c is the cutoff radius of the potential then among all pairs formed by an atom *i* in the box and the set of all periodic images of atom *j* atmost one will interact. To demonstrate this, if suppose an atom *i* interacts with two images j_1 and j_2 of an atom *j*. These two images must be separated by a distance of at least 2 r_c. In order to interact with both j_1 and j_2, *i* should be within a distance r_c from each of them, which is impossible since they are separated by more than 2 r_c. Using the minimum image criterion, only the closest image of an atom *j* if within the cutoff radius r_c will interact with an atom *i*. Hence, it is very important that the box size of the simulation cell is at least greater than twice the cutoff radius specified for the potential. Long range interactions such as Columbic interactions larger than the simulation cell are treated using Ewald summation which was developed for studying ionic crystals (Ewald 1921; Madelung 1918). Using the minimum image criterion, the periodic boundaries are modeled by modifying the distance calculation. The difference between the x- coordinate of an atom *i* and *j* is denoted as *dx*, then if *dx* is greater than half the box size in x- direction (L/2) then atom *j* does not interact with atom *i.* Instead, its image j_1 in a surrounding cell is considered to be bonded with atom *i.*

If, dx > L/2 then dx = dx – L,
If, dx < -L/2 then dx = dx + L. (3.14)

Alternatively,

$$dx = dx \; L \; \text{anint}\left(\frac{dx}{L}\right), \tag{3.15}$$

where L is the box size in that particular Cartesian direction and anint is a function that rounds off the ratio $\frac{dx}{L}$ to the nearest integer value.

Similar procedure is applied in the Y- and Z-directions, and the distance is computed as:

$$\sqrt{dx^2 + dy^2 + dz^2}\ . \tag{3.16}$$

CHAPTER 4

Interatomic Potentials

The interatomic potential used in a simulation to model the lattice structure plays an important role in determining the accuracy of the simulation results. The accuracy of the trajectories obtained from MD simulation is affected by the choice of the potential energy function. The total energy of the system is the sum of kinetic energy (KE) and the potential energy (PE). The KE is simple to compute but the PE computation is complex since it depends on the positions of all interacting atoms. The complexity of the potential also determines the computational time required for the simulation. The force acting on an atom is proportional to the first derivative of the potential energy function. The largest part of a molecular dynamics simulation is the evaluation of the forces which are required to calculate new positions of atoms. The potential energy can be obtained using two approaches. One approach is to perform *ab initio* calculations by solving the Schrodinger's equation for the entire system at each integration step. Although, theoritically this is the best approach (without any arbitrary assumptions and *ad hoc* parameterization), it is not feasible to perform such calculations for the entire system at every time step. For larger systems comparitively simple and computationally efficient empirical potential functions are used which take into account factors such as, bond stretching, bending and torsions, covalent bonding and non-bonded (Van der Waals and Coulomb) interactions.

4.1 Design of inter-atomic potentials

The traditional approach is to completely get rid of electronic degrees of freedom and move the nuclei using some appropriate analytical potential function. The functional form of this potential is chosen so as to mimic the behavior of the true potential in realistic ways for specific materials under specific conditions. Constructing the potential involves two steps:

1. Selecting an analytical form for the potential. This could be sums of pairwise terms where the energy of the pair depends on the relative distance, or a many-body form appropriate for specific type of bonding involving bond distances, angles, and coordination.
2. Once the form is decided then the next step is to determine specific parameterization of the functions. A set of parameters is found which fits the data from *ab initio* calculations or experimental properties, such as, density, cohesive energy and phonon frequencies, or a data set containing some combination of both theoretical and experimental observations.

The most important factor is the range of applicability of the potential energy function. Since *ad hoc* functional form is used, it has no theoretical foundation and once the empirical potential is developed there is no straightforward means to improve it. Normally, the parameters are adjusted to the equilibrium data which means that the potential can be used to model the bulk properties of the crystal lattice close to equilibrium conditions. So it is unlikely that the energies and forces for amorphous structures will be accurately represented by the potential. For example, a potential function designed for bulk may not be appropriate for studying the bond dissociation or

diatomic molecule of the same element, since the environment would be very different from the one for which it was designed. However, it may be possible to model the bulk and surface where the environment differs due to the reduced coordination of the atoms at the surface. These problems are intractable and one should be critical while using analytical potentials for simulating high temperature and pressure conditions.

4.2 Classification of potential energy functions

The empirical potentials are further classified into two-body, three-body and many-body potential forms. Most of the empirical potential forms comprise an attractive and a repulsive term. The term *empirical* may be misleading as it may not be strictly empirical as the term suggests (Komanduri and Raff, 1999). These potentials can provide a more realistic model than the potentials derived from purely theoretical considerations (Torrens, 1972) since parameterization of these potentials can take into account some physical properties as well.

Empirical potential forms are based on mathematical expressions for pairwise interaction between two atoms or ions, which may or may not be justified from theory. These forms can be expressed in terms of attractive and repulsive terms. By convention repulsive forces are considered positive and attractive forces as negative. As the distance between two atoms decreases, both attractive and repulsive forces increase. The repulsive forces increase more rapidly than the attractive forces. The curvature of the potential energy function is mainly determined by the repulsive force component, which in turn also governs the elastic behavior of the solid. When attractive and repulsive forces exactly balance each other, this corresponds to the equilibrium position with the potential energy being minimum, which is also the bond energy. The cohesive properties of solids,

such as melting and vaporization behavior are determined by the magnitude of the maximum binding energy, which is governed by attractive force component. Higher the bonding energy, the higher is the melting temperature and the elastic modulus and lower is the coefficient of thermal expansion.

Several empirical potential functions have been developed suitable for different materials. Some examples of simple pair potentials are Lennard- Jones (also known as 6-12) (1925), Morse potential (1929) and Born-Meyer potential (1933). Also for complex systems such as one involving covalent bonding common potential functions are Stillinger-Weber (1985), Tersoff (1988, 1989), Bolding-Anderson potential (1990), Brenner potential (1990), and embedded atom potential and modified emebedded atom potential (MEAM) (Baskes, 1989).

Cutoff radius

In general, each atom can interact with all other atoms in the simulation. One method of increasing the computational speed is to limit the range of the potential. Normally the potential functions are truncated at a certain value of bond distance called the "cutoff radius". By neglecting the interaction beyond the cutoff radius, the computational time is significantly reduced with very little loss in accuracy. Truncation of the potential energy function also results in the truncation of the forces. The cutoff distance is generally taken as the distance where the potential energy is ~3-5% of the equilibrium potentials energy. Consequently, a finite cutoff results in small fluctuations in total energy instead of strictly constant energy.

4.2.1 Pair potentials

Pair potentials are applicable for many metals for atomistic studies. Pair potentials can be classified into two basic categories (Vitek, 1996), one in which the potential determnies the total energy of the system and the second type determines the change in energy when a configuration varies under constant density conditions.

Lennard- Jones potential (Lennard- Jones, 1925) is given as:

$$V_{LJ}(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right], \tag{4.1}$$

where ε and σ are parameters that determine the well depth and the position of the potential minimum, respectively. One of the most commonly used forms of the pair potential is the Morse potential (Morse, 1929). It is used to model the interaction between atoms of two different materials. It is given as:

$$V(r) = D\left(1 - e^{-\alpha(r-r_e)}\right)^2,$$
$$\text{or } V(r) = D\left(e^{-2\alpha(r-r_e)} - 2e^{-\alpha(r-r_e)}\right) \ldots r \le r_c. \tag{4.2}$$

where r is the bond distance between the two atoms, and r_c is the cutoff radius beyond which the potential is truncated to zero. Potential values computed using the two forms given in Eqn. (4.2) differes by a factor of D, which is a constant, and as a result the forces computed using the two forms are the same and either of the forms could be used in simulation. The potential parameters D, α, r_e are adjusted to the measured sublimation enthalpy, Debye temperature, and the nearest neighbor spacing for the material. Garifalco and Weizer (1959) calculated Morse potential parameters using experimental values for energy of vaporization, lattice constants and compressibility. Morse potential can model FCC metals well but not BCC metals and covalent materials. A general feature of a

physically reasonable pair potential is that they favor formation of closed packed structures. So they are unsuitable for describing covalent systems which assume more open structure.

4.2.2 Many-body potentials

Pair potentials, such as Lennard-Jones potential are suitable to model interactions of inert gases, such as Helium, Argon, and Xenon. A covalent material represents a difficult challenge because of complex quantum mechanical effects, such as chemical bond formation, hybridization, metallization, charge transfer, and bond bending, which must be described by an effective interaction between atoms. For instance, silicon undergoes a series of structural phase transitions from tetrahedral to a denser phase called as β-tin to simple cubic to FCC under pressure. This indicates that the energy difference between these structures is not too large which suggests that the cohesive energy is nearly independent of the coordination. The pairwise potential model would favor the more packed structure. Consequently, no reasonable pair potential can stabilize the diamond tetrahedral structure without considering the angular terms. Various methods have been introduced to circumvent this problem. Several authors have introduced potentials with a strong angular dependence (Stillinger and Weber, 1985; Biswas and Hamman, 1985; Baskes, 1987; Baskes et al, 1989). Pettifor (1989) developed similar approach from tight binding. Models based on the bond charge (Brenner and Garrison, 1985), and a function of local coordination (Tersoff, 1986) have also been developed. Most of these models have been successful in the regime for which they were parameterised but have shown a lack of transferability. A survey of six such potentials (Balamane and Halicioglu, 1992) concluded that each has strengths and limitations, none appearing clearly superior to

others, and none being fully transferable. Recently the environment dependent interatomic potential (EDIP) (Justo et al, 1997, 1998; Bazant et al, 1997) have been developed as an improvement over previous model.

The other group of potentials (Kane, 1985; Keating, 1966) are constructed to accurately describe small distortions from the ground state in complex systems, such as diamond structure of semiconductors. The Keating model uses Taylor series expansion of the energy about its minimum. It can give accurate descriptions of small displacements but become progressively less accurate for large displacements. Such potentials are useful for describing phonons and elastic deformations.

In general, any many-body potential energy function describing interactions among *N* identical particles can generally be resolved into one-body, two-body, three-body etc. contributions as follows:

$$E = \sum_{i} V_1(i) + \sum_{\substack{i,j \\ i<j}} V_2(i,j) + \sum_{\substack{i,j,k \\ i<j<k}} V_3(i,j,k) + \ldots + V_N(1,\ldots,N). \tag{4.3}$$

The single particle potential V_1 describes external forces. The first term which describes the interactions of atoms is the second term, which has a form of pair potential. In order that this description to be useful in theoretical modeling it is necessary that the component functions V_n quickly converges to zero with increasing *n*.

The first step towards a many-body form is to include three-body terms, by favoring the bond angles corresponding to those of diamond structure. Stillinger and Weber (1985) proposed such a potential which has the form:

$$V = \frac{1}{2}\sum_{ij} \Phi(r_{ij}) + \sum_{ijk} g(r_{ij}) \cdot g(r_{ik}) \cdot \left(\cos(\theta_{ijk}) + \frac{1}{3} \right)^2, \tag{4.4}$$

where θ_{ijk} is the bond angle formed by the bonds *ij* and *ik* and *g(r)* is a decaying function with a cutoff between the first and second neighbor. The bond angles in diamond

tetrahedral structure are ~ 109.5° and the cosine of this angle is ~ -1/3. This makes the term $cos(\theta_{ijk})+1/3$ to go to zero for θ ~ 109.5°, which makes tetrahedral structure more stable than compact structure. This potential gives a fairly realistic description of crystalline silicon. However its built-in tetrahedral bias create problems in other situations for example, it cannot predict the right energies of the non-tetrahedral structures found under pressure, the coordination of the liquid is too low and the surface structures are not correct. Consequently this potential is not transferable to situations other than it was designed for.

Work by Carlsson (1990) showed that to build realistic potential, analytic forms which take into account local environments with bond strengths depending on it should be considered. The work of Biswas and Hamman (1987) suggests that a three-body potential is not adequate for accurately describing the cohesive energy of silicon over a wide range of bonding geometry and coordination. However a general form for a four-body or five-body potential becomes intractable as it would contain too many parameters. Instead of a general *N*-body form, Tersoff (Tersoff, 1988; 1989) proposed an approach by coupling two body and multi body correlations into the model. The potential is based on bond order which takes into account local environment. The bond strength depends on geometry, the more neighbors an atom has, the weaker the bond to each neighbor would be. If the energy per bond decreases rapidly with increasing coordination then the diatomic molecules would be the most stable arrangements of atoms. On the other hand, if the bond order depends weakly on coordination then this should favor closed packed structures to maximize the number of bonds. Silicon undergoes structural transitions with a varying coordination under pressure with a small difference in cohesive energy (Yin

and Cohen, 1980, 1982, 1984; Chang and Cohen, 1984). This suggests that the decrease in bond strength with increase in coordination number almost cancels the increase in number of bonds over a range of coordination (Tersoff, 1988).

Tersoff potential (Tersoff, 1989) is given as:

$$E = \sum_i E_i = \frac{1}{2}\sum_{i \neq j} V_{ij}, \qquad (4.5\text{ a})$$
$$V_{ij} = f_C(r_{ij})\left[f_R(r_{ij}) + b_{ij}\ f_A(r_{ij})\right].$$

where E is the total potential energy of the system. f_R and f_A are repulsive and attractive pair potentials, respectively, and f_C is a cutoff function given by:

$$f_R(r_{ij}) = A_{ij}\, e^{(-\lambda_{ij} r_{ij})}, \qquad (4.5\text{ b})$$
$$f_A(r_{ij}) = -B_{ij}\, e^{(-\mu_{ij} r_{ij})},$$

$$f_C(r_{ij}) = \begin{cases} 1, r_{ij} < R \\ \frac{1}{2} + \frac{1}{2}\cos\left[\pi \frac{(r_{ij} - R)}{(S - R)}\right], R < r_{ij} < S \\ 0,\ r_{ij} > S \end{cases}$$

where r_{ij} is the bond distance between atoms i and j. S is the cutoff radius. λ_{ij}, μ_{ij}, and R are potential parameters. The main feature of this form is the term b_{ij}. The idea is that the strength of each bond depends upon the local environment and is lowered when the number of neighbors is relatively high. This dependence is expressed by the parameter b_{ij} which can diminish the attractive force relative to the repulsive force.

$$b_{ij} = \chi_{ij}(1 + \beta_i^{n_i}\ \zeta_{ij}^{n_i})^{-1/2n}, \qquad (4.5\text{ c})$$
$$\zeta_{ij} = \sum_{k \neq i,\, j} f_C(r_{ik})\, \omega_{ik}\ g(\theta_{ijk}),$$
$$g(\theta_{ijk}) = 1 + \frac{c_i^2}{d_i^2} - \frac{c_i^2}{\left[d_i^2 + (h_i - \cos(\theta_{ijk}))^2\right]},$$

The term ζ_{ij} defines the effective coordination number of atom taking into account the relative distance between two neighbors $(r_{ij}\text{-}r_{ik})$ and the bond angle θ_{ijk}. The function $g(\theta)$

has a minimum at $h=cos(\theta)$, the parameter d determines how sharp the dependence on the angle is and c expresses the strength of the angular effect. The parameters R and S are not optimized but chosen so as to include first neighbors only for several selected high symmetry structures, such as: graphite, diamond, simple cubic, and face centered cubic. The parameter x_{ij} strengthens or weakens heteropolar bonds in multi-component systems. x_{ii} =1, and $x_{ij} = x_{ji}$. Also ω_{ii} = 1. The potential was first calibrated for silicon (Tersoff, 1988)[a] and then for carbon (Tersoff, 1988)[b]. The parameters were chosen to fit theoretical and experimental data, such as cohesive energy, lattice constants, and bilk modulus obtained from realistic and hypothetical configurations. Table 4.1 lists the parameters for carbon, silicon and germanium.

Table 4.1 (Parameters of Tersoff potential, 1989)

	C	Si	Ge
A (eV)	1.3936×10^{3}	1.8308×10^{3}	1.769×10^{3}
B (eV)	3.467×10^{2}	4.7118×10^{2}	4.1923×10^{2}
λ ($°A^{-1}$)	3.4879	2.4799	2.4451
μ ($°A^{-1}$)	2.2119	1.7322	1.7047
β	1.5724×10^{-7}	1.1×10^{-6}	9.0166×10^{-7}
n	7.2751×10^{-1}	7.8734×10^{-1}	7.5627×10^{-1}
c	3.8049×10^{4}	1.0039×10^{5}	1.0643×10^{5}
d	4.384	1.6217×10^{1}	1.5652×10^{1}
h	-5.7058×10^{-1}	-5.9825×10^{-1}	-4.3884×10^{-1}
R (°A)	1.8	2.7	2.8
S (°A)	2.1	3.0	3.1

Brenner (1990) developed an empirical many-body potential for hydrocarbons which can model intramolecular chemical bonding in a variety of small hydrocarbon molecules as well as graphite and diamond lattice. The potential function was based on Tersoff covalent bonding formulation, with additional terms to correct an inherent overbinding of radicals. Nonlocal effects were incorporated via an analytic function that defines conjugation based on the coordination of carbon atoms. The potential was fitted

to the binding energy of the C_2 diatomic molecule, and binding energies and lattice constants of graphite, diamond, simple cubic and face centered cubic structures. Dyson and Smith (1996) extended the Brenner potential for C-Si-H system to model the chemical vapor deposition (CVD) diamond growth on the silicon substrate. Parameters for the Si-Si and Si-C interactions were fitted to diatomic bond energy and bulk properties such as bulk modulus, experimental cohesive energy, experimental lattice constant instead of bond length, phonon mode frequencies.

CHAPTER 5

Ab Initio Calculations

'*Ab initio*' is a term used to describe a solution of the non-relativistic, time independent Schrödinger equation: $\hat{H}\Psi = E\Psi$. Where $\hat{H}$ is the Hamiltonian operator for the system, which is a function of kinetic and potential energies of the particles of the system, Ψ is the wavefunction, and E is the energy.

The Hamiltonian for an *N* electron atom is

$$H = -\frac{\hbar^2}{2m_n}\nabla_n^2 - \frac{\hbar^2}{2m_e}\sum_{i=1}^{N}\nabla_i^2 - \sum_{i=1}^{N}\frac{Z_e^2}{r_i} + \sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{e^2}{r_{ij}}. \qquad (5.1)$$

The first term on the right hand side of the Eqn.(5.1) represents the kinetic energy of the nucleus. It contains the coordinates of the nucleus and derivatives with respect to the coordinates of the nucleus, but it does not contain the coordinates of any of the *N* electrons or derivatives with respect to these coordinates. Therefore, this is called as zero-electron term. The second and third terms in Eqn.(5.1) contains the coordinates of one electron and hence are called as one electron terms, where as the last term, electron-electron repulsion, depends on coordinates of both electrons and hence is called as two electron terms.

5.1 Born-Oppenheimer approximation

It is computationally impossible to solve Eqn.(5.1) for anything other than the hydrogen atom. For this reason, for a many bodied system, approximations are introduced to simplify the calculations. One of the approximations is the *Born-Oppenheimer* approximation. It can be noted that the mass term appears in the denominator of the nuclear kinetic energy term is much larger than the mass of the electron. This large mass means that the nuclei are moving very slowly relative to the electrons. The nuclei can be considered to be moving in the potential generated by the moving electrons. This approximation allows the electronic Hamiltonian to be considered for a fixed set of nuclear co-ordinates.

5.2 Basis sets

The next approximation involves expressing the molecular orbitals as linear combinations of a pre-defined set of one-electron functions known as basis functions. A basis set is the mathematical description of the orbitals within a system, which in turn combine to approximate the total electronic wavefunction. These basis functions are usually centered on the atomic nuclei and so bear some resemblance to atomic orbitals.

An individual molecular orbital is defined as:

$$\Phi_i = \sum_{\mu=1}^{N} c_{\mu i}\, \chi_\mu \,. \tag{5.2}$$

The coefficients $c_{\mu i}$ are known as the molecular orbital expansion coefficients. The basis functions $\chi_1 \ldots \chi_N$ are also chosen to be normalized.

5.2.1 Slater-type orbitals (STO)

STOs are constructed from a radial part describing the radial extend of the orbital and an angular part describing the shape of the orbital.

$$\Phi_{\mu} = C\, r^{n-1}\, e^{(-\zeta r)}\, Y_{lm}. \qquad (5.3)$$

The radial part $r^{n-1}\, e^{(-\zeta r)}$ depends on the distance r from the origin of the basis function (usually the location of the nucleus), the orbital exponent, and the principal quantum number, *n*. The spherical part, Y_{lm} known as spherical harmonics, depends on the angular quantum number, *l* and the magnetic quantum number, *m*. The normalization constant, C is chosen such that the integral over the square of the basis function yields unity.

5.2.2 Gaussian-type orbitals (GTO)

Gaussian-type orbitals are constructed from a radical and a spherical part, but the radical part has a different dependence on *r*. The GTO squares *r* so that the product of gaussian primitives is another gaussian. By doing this, the equation is much easier at the cost of accuracy. To compensate for this, more gaussian equations are combined to get more accurate results. Gaussian and other *ab initio* electronic structure programs use gaussian type atomic functions as basis functions. Gaussian functions have a general form as:

$$g = c\, x^{a}\, y^{b}\, z^{c}\, e^{-\alpha r^{2}}, \qquad (5.4)$$

where α is the constant determining the size (radical extent) of the function. In a Gaussian function $e^{-\alpha r^{2}}$ is multiplied by powers of x, y, z, and normalization constant so that:

$$\int_{all\ space} g^{2} = 1. \qquad (5.5)$$

The sum of these exponents L= a+ b+ c is used to define the angular momentum of the basis function: s- type (L=0), p- type (L=1) d- type (L=2), f- type (L=3), g- type (L=4)…

Following are some representative Gaussian functions for *s*, p_y and d_{xy} respectively:

$$g_s(\alpha, r) = \left(\frac{2\alpha}{\pi}\right)^{3/4} e^{-\alpha r^2}, \tag{5.6}$$

$$g_y(\alpha, r) = \left(\frac{128\alpha^5}{\pi^3}\right)^{1/4} y\, e^{-\alpha r^2},$$

$$g_{xy}(\alpha, r) = \left(\frac{2048\alpha^7}{\pi^3}\right)^{1/4} xy\, e^{-\alpha r^2}.$$

Linear combinations of primitive Gaussians are used to form the actual basis functions called *contracted Gaussians* which have the form:

$$\chi_\mu = \sum_p d_{\mu p}\, g_p\,, \tag{5.7}$$

where g_p are primitive Gaussians, $d_{\mu p}$ are fixed constants within a given basis set.

This results in the following expansion for molecular orbitals:

$$\Phi_i = \sum_\mu c_{\mu i}\, \chi_\mu = \sum_\mu c_{\mu i} \left(\sum_p d_{\mu p}\, g_p \right). \tag{5.8}$$

5.2.3 Types of basis sets

Larger basis sets more accurately approximate the orbitals by imposing fewer restrictions on the locations of electrons in space. In the true quantum mechanical picture, electrons can have a finite probability of existing anywhere in space; this limit corresponds to the infinite basis set expansion. Standard basis sets for electronic calculations use linear combinations of Gaussian functions to form orbitals. Basis sets may be classified by the number and type of basis functions that they contain. Basis sets

assign a group of basis functions to each atom within a molecule to approximate its orbitals. These basis functions are composed of a linear combination of Gaussian functions referred to as *contracted* functions, and the component Gaussian functions are referred to as *primitives*. A basis function consisting of a single Gaussian function is termed as *uncontracted*.

Minimal basis set

Single Gaussian functions as described by Eqn.(5.4) are not well suited to describe the spatial extent and nodal characteristics of atomic orbitals. Hence, basis functions are described as sum (contraction) of several Gaussians as used in Eqn.(5.7). Minimal basis set contains minimum number of basis functions needed for each atom. Minimal basis set use fixed size atomic type orbitals. The STO-3G basis set (Hehre *et al.*, 1969; Collins *et al.*, 1976) is a minimal basis set (although it is not the smallest possible basis set). It uses three Gaussian primitives per basis function, which accounts for *'3G'* in its name. *'STO'* stands for Slater-type orbitals. The STO-3G basis set approximates Slater-type orbitals with Gaussian functions.

Double zeta, triple zeta and quadruple zeta basis set

The minimal basis set approximates all orbitals to be of the same shape. The *double zeta* basis sets such as the Dunning-Huzinaga basis set denoted as D95 (Dunning and Huzinaga, 1976) from all molecular orbitals from linear combinations of two sizes of functions for each atomic orbital. Dunning's basis set has been derived from an already existing large atomic basis set of nine uncontracted Gaussian primitives of the s-type and five uncontracted Gaussian primitives of the p-type. Six of the nine s-type functions have then been grouped into a single contraction, while the other three s-type functions have

been left alone. Similarly, four of the five p-type functions have been contracted into a single function, while one function was left uncontracted. This yields a basis set of four s-type and two p-type basis functions. In contrast to the split valence basis sets discussed before, the D95 basis set is a full double zeta basis set in that it allocates two basis functions for each atomic orbital of the core as well as the valence region occupied in the electronic ground state.

The *double zeta* basis set is important because it allows to treat each orbital separately during Hartree-Fock calculations. This gives more accurate representation of each orbital. Each atomic orbital is expressed as a sum of two Slater -type orbitals. The two equations are the same except for the value of *zeta*, which accounts for how diffuse (large) the orbital is. The two STOs are added in some proportion as in Eqn.(5.9).

$$\Phi_{2s} = \underbrace{\Phi_{2s}^{STO}(r,\zeta_1)}_{\text{Slater- orbital 1}} + \underbrace{d \cdot \Phi_{2s}^{STO}(r,\zeta_2)}_{\text{Slater- orbital 2}} \qquad (5.9)$$

Constant

The constant *d* determines how much each STO will account towards the final orbital. Thus, the size of the atomic orbital can range anywhere between the value of either of the two STOs. The triple and quadruple zeta basis sets use three and four Slater equations respectively.

Split valence basis set

The description of the valence electrons can be improved over that in the minimal STO- 3G basis set if more than one basis function is used per valence electron. Basis sets of this type are called *split valence basis set*. They have two sizes of basis function for each valence orbital. Often, it takes too much effort to calculate a double-zeta for every

orbital. Since the inner-shell electrons are not as vital to the calculation, they are described with a single Slater Orbital and the only valence orbitals are treated using a double-zeta. Examples of split valence basis sets are 3-21G (Binkley *et al.,* 1980; Gordon *et al.,* 1982; Pietro *et al.,* 1982; Dobbs and Hehre, 1986; 1987) and 6-31G (Ditchfield *et al.,* 1971; Hehre *et al.,* 1972; Hariharan and Pople, 1973; 1974; Gordon, 1980; Binning and Curtiss, 1990). The notation 3- 21G means summing 3 Gaussians for the inner shell orbital, each valence electron is described by two basis functions, two Gaussians for the first STO of the valence orbital and 1 Gaussian for the second STO.

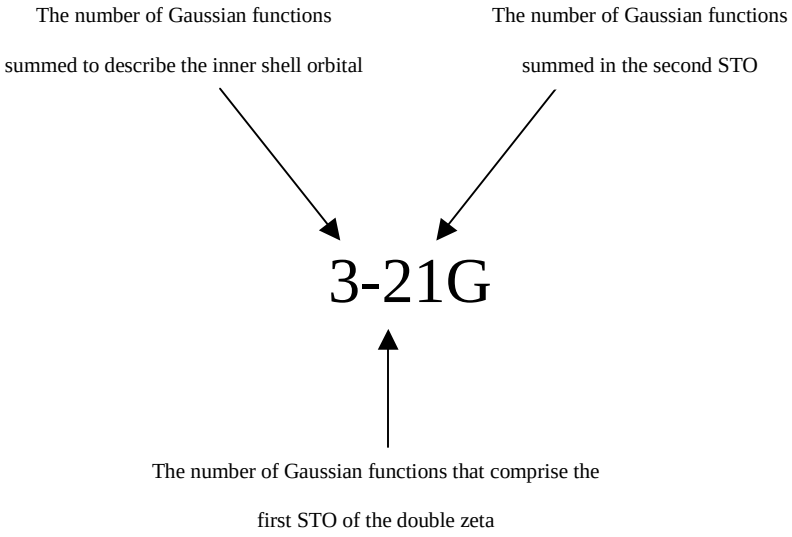


Fig 5.1 Description of split valence basis set

The main difference between 3-21G and 6-31G is that a much larger number of primitives are used in the latter in the core as well as the inner most valence shell. The use of a contraction of six Gaussian primitives for each core orbital improves the description of the core region significantly. The valence region is described by two basis

functions per atomic orbital. The inner shell is composed of a contraction of three Gaussians and the outer shell consists of one single Gaussian primitive.

6-31G† also known as 6-31G(d′) and 6-31G†† also known as 6-31G(d′,p′) is defined as part of the complete basis set methods (Petersson and Al-Laham, 1991 ; Petersson *et al.* 1988). Similarly, *triple split valence* basis sets such as 6-311G also known as MC-311G (McLean and Chandler, 1980; Krishnan *et al.,* 1980; Wachters, 1970; Hay, 1977; Raghavachari and Trucks, 1989; Binning and Curtiss, 1990; Curtiss *et al.,* 1995; McGrath and Radom, 1991) use three sizes of contracted functions for each orbital type.

Polarized basis set

A better approximation is to account for the fact that sometimes orbitals share qualities of 's' and 'p' orbitals or 'p' and 'd', etc. and not necessarily have characteristics of only one or the other. As atoms are brought close together, their charge distribution causes a polarization effect (the positive charge is drawn to one side while the negative charge is drawn to the other) which distorts the shape of the atomic orbitals. Split valence basis sets allow orbitals to change size, but not to change shape. Polarized basis sets remove this limitation by adding orbitals with angular momentum beyond what is required for the ground state to the description of each atom. The first step consists of the addition of a set of d-type functions to the basis sets of those atoms, which have occupied s- and p-shells in their electronic ground states. For hydrogen, this corresponds to the addition of a set of p-type functions. Two different notations exist to specify the addition of polarization functions. The first notation adds one asterisk to the basis set to specify addition of polarization functions to non-hydrogen atoms, while two asterisks symbolize

the addition of polarization functions to all atoms (including hydrogen). The 6-31G** basis set (Frisch *et al.,* 1984) is thus constructed from the split valence 6-31G basis set through addition of one set of d-functions to all non-hydrogen atoms and one set of p-functions to all hydrogen atoms. In the second notation the polarization functions are specified through their angular quantum number explicitly. The 6-31G** basis set would then be termed 6-31G (d,p). This latter notation is much more flexible as multiple sets of polarization functions can be specified much more easily. The notation 6-31G (d) means the polarization functions for the 6-31G basis appear in the basis set listing as a single set of uncontracted d-type Gaussians. There are six Cartesian d-type Gaussians (x^2, y^2, z^2, xy, yz, zx) $e^{(-\alpha r2)}$, which are equivalent to five pure d-type functions (xy, yz, zx, x^2-y^2, $3z^2$-r^2) $e^{(-\alpha r2)}$ plus one additional s-type function.

Diffuse functions

In chemistry, one is mainly concerned with the valence electrons which interact with other molecules. However, many of the basis sets concentrate on the main energy located in the inner shell electrons. This is the main area under the wave function curve. In Fig 5.2, this area is that to the left of the dotted line. Normally the tail (the area to the right of the dotted line), is not really a factor in calculations.

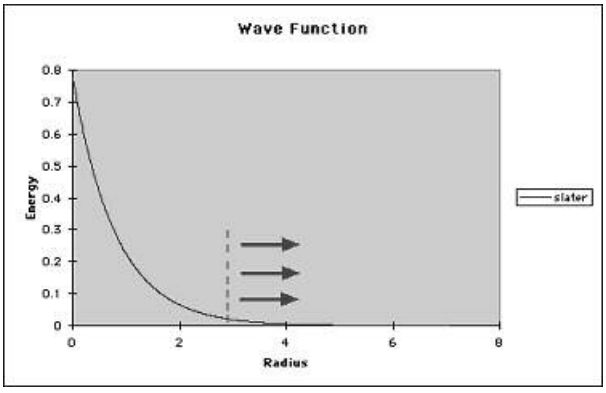


Fig 5.2 Variation of energy with respect to radius

However, when an atom is in an anion or in an excited state, the loosely bond electrons, which are responsible for the energy in the tail of the wave function become much more important. The theoretical description of negatively charged species is particularly challenging for *ab initio* molecular orbital theory. This is due to the fact that the excess negative charge spreads outward to a much larger degree than is typically the case for uncharged or positively charged molecules. To compensate for this, diffuse functions are used. Diffuse functions are large size versions of s- and p-type functions (as opposed to standard valence size functions). They use small orbital exponents. They allow orbitals to occupy a larger region of space and are important for systems where electrons are relatively far from the nucleus, such as molecules with lone pairs, anions and other systems with significant negative charge, systems in their excited states, and systems with low ionization potentials. Diffuse basis functions are typically added as an additional set of uncontracted Gaussian functions of the same angular momentum as the valence electrons. To reflect the addition of diffuse basis functions on all non-hydrogen atoms, a + sign is added to the standard basis set notation. If diffuse s-type functions are also added to the basis set of hydrogen atoms, a second + sign is appended. The 6-31+G(d) basis set (Clark *et al.,* 1983) is the 6-31G(d) basis set with diffuse functions added to heavy atoms. The basis et 6-31++G(d) adds diffuse functions to the hydrogen as well. Diffuse functions on hydrogen seldom make a significant difference in accuracy.

High angular momentum basis set

Such basis sets add multiple polarization functions per atom to the triple zeta basis set. For example, the 6-31G(2d) basis set adds two *d* functions per heavy atom instead of just one. Such basis sets are useful for describing the interactions between

electrons in electron correlation methods; they are not generally needed for Hartree-Fock calculations, to be discussed in the next section.

Basis sets for post- third row atoms

basis sets for atoms beyond the third row of the periodic table are handled somewhat differently. For very large nuclei, electrons near the nucleus are treated in an approximate way, using effective core potentials (ECP). This treatment includes relativistic effects. The LANL2DZ basis set is an example of such a basis set.

5.3 Hartee-Fock method

Molecular orbital theory decomposes the wave function Ψ into a combination of molecular orbitals Φ_1, Φ_2… Φ_i are chosen to be normalized orthogonal set of molecular orbitals. Thus

$$\int\int\int \Phi_i^* \Phi_i \, dx\, dy\, dz = 1, \tag{5.10}$$

$$\int\int\int \Phi_i^* \Phi_j \, dx\, dy\, dz = 0;\ i \neq j\,. \tag{5.11}$$

The simplest way of expressing the wave function of a many electron system is to take the product of the spin-orbitals of the individual electrons. For the case of two electrons

$$\Psi(x_1 x_2) = \chi_1(x_1)\,\chi_2(x_2)\,. \tag{5.12}$$

This is known as Hartree product.

However, such a function is not antisymmetric, since swapping the orbitals of two electrons does not result in a sign change. i.e.

$$\Psi(x_1 x_2) \neq -\Psi(x_2 x_1)\,. \tag{5.13}$$

Therefore the Hartree product does not satisfy the Pauli principle. This problem can be overcome by taking a linear combination of both Hartree products:

$$\Psi(x_1 x_2) = \frac{1}{\sqrt{2}}\{\chi_1(x_1)\chi_2(x_2) - \chi_1(x_2)\chi_2(x_1)\}, \quad (5.14)$$

where the coefficient is the normalization factor. This wave function is antisymmetric. Moreover, it also goes to zero if any two wave functions or two electrons are the same. The expression can be generalized by writing it as a determinant called Slater determinant:

$$\Psi(x_1, x_2, ..., x_N) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \chi_1(x_1) & \chi_2(x_2) & ... \chi_N(x_1) \\ \chi_1(x_2) & \chi_2(x_2) & ... \chi_N(x_2) \\ . & . & ... \quad . \\ . & . & ... \quad . \\ . & . & ... \quad . \\ \chi_1(x_N) & \chi_2(x_N) & ... \chi_N(x_N) \end{vmatrix} \quad (5.15)$$

A single Slater determinant is used as an approximation to the electronic wave function in Hartree-Fock theory. In more accurate theories, such as configuration interaction and multiconfiguration self consistent field (MCSCF) calculation (Hegarty and Robb, 1979; Eade and Robb, 1981; Schlegel and Robb, 1982; Bernardi, *et al.*, 1984; Frisch *et al.*, 1992), a linear combination of Slater determinants is used. Each row is formed by representing all possible assignments of electron *i* to all orbital spin combinations. Swapping two electrons corresponds to interchanging two rows of the determinant, which will have the effect of changing its sign.

The original Hartree method expresses the total wave function of the system as a product of one-electron orbital. In the Hartree-Fock method (Hehre *et al.*, 1986), the wave function is an antisymmetrized determinantal product of one-electron orbital (the "Slater" determinant). Schrödinger's equation is transformed into a set of Hartree-Fock equations. Now the problem is to solve for a set of molecular orbital expansion

coefficients $c_{\mu i}$ in Eqn.(5.8). Hartree-Fock theory takes advantage of the variational principle, which states that for the ground state of any antisymmetric normalized function of the electronic coordinates denoted as Ξ, the expectation value corresponding to Ξ will always be greater than the energy for the exact wave function:

$$E(\Xi) > E(\Psi). \tag{5.16}$$

In other words, the energy of the exact wave function serves as a lower bound to the energies calculated by any other normalized antisymmetric function. Thus the problem becomes one of finding the set of coefficients that minimize the energy of the resultant wave function.

The variables in the Hartree-Fock equations depend on themselves. So, they must be solved in an iterative manner. Convergence may be improved by changing the form of the initial guess. Since the equations are solved self-consistently, Hartree-Fock is an example of a self-consistent field (SCF) method.

Hartree-Fock calculation involves the following steps:

1. Begin with a set of approximate orbital for all the electrons in the system.
2. One electron is selected, and the potential in which it moves is calculated by freezing the distribution of all the other electrons and treating their averaged distribution as the Centro-symmetric source of potential
3. The Schrödinger equation is solved for this potential, which gives a new orbital for it.
4. The procedure is repeated for all the other electrons in the system, using the electrons in the frozen orbitals as the source of the potential.
5. At the end of one cycle, there are new orbitals from the original set.

6. The process is repeated until there is little or no change in the orbitals.

Unrestricted Hartree-Fock method is capable of treating unpaired electrons for open shell systems. In this case the α and β electrons are in different orbitals, resulting in two sets of molecular orbital expansion coefficients.

$$\phi_i^{\alpha} = \sum_{\mu} c_{\mu i}^{\alpha} \chi_{\mu},$$
$$\phi_i^{\beta} = \sum_{\mu} c_{\mu i}^{\beta} \chi_{\mu}.$$

The two sets of coefficients result in two sets of Fock matrices producing two sets of orbitals. These separate orbitals proper dissociation to separate atoms and correct delocalized orbitals for resonant systems. However, the eigenfunctions are not pure spin states, but contain some amount of *spin contamination* from higher states, for example, doublets are contaminated to some degree by functions corresponding to quartets and higher states..

It might give an impression that molecular orbitals are real. Except for the hydrogen atom that is not the case. Molecular orbitals are the product of Hartree-Fock theory, which is an approximation to the Schrödinger equation. The approximation is that each electron feels only the average Coulomb repulsion of all the other electrons. This approximation makes Hartree-Fock theory much simpler than the real problem, which is an *N*-body problem. Unfortunately, in many cases this approximation is rather serious and can give bad answers. It can be corrected by explicitly accounting for the electron correlation by many-body perturbation theory (MBPT), configuration interaction (CI), or density functional theory (DFT).

5.4 Electron correlation methods

Hartree-Fock theory provides an inadequate treatment of the correlation between the motions of electrons within a molecular system, especially that arising between electrons of opposite spin. When Hartree-Fock theory fulfills the requirement that $|\Psi|^2$ be invariant with respect to the exchange of any two electrons by antisymmetrizing the wave function, it automatically includes the major correlation effects arising from pairs of electrons with the same spin. This correlation is termed as *exchange correlation.* The motion of electrons of opposite spin remains uncorrelated under Hartree-Fock theory. Any method which goes beyond SCF in attempting to treat this phenomenon is known as *electron correlation* method or *post- SCF* method.

5.4.1 Requirements of electron correlation theories

Any electron correlation theory should provide a unique total energy for each electronic state at a given geometry and should also provide continuous potential energy surfaces as the geometry changes. The resulting energy should be variational, i.e. it should be an upper bound to the exact energy. The most important criterion for an accurate electron correlation theory is the property of size consistency. This term refers to the linear scaling of energy with the number of electrons in the system. A size consistent method leads to additive energies for infinitely separated systems. Size consistent method is necessary to reach quantitative accuracy. Another important criterion is the correctness for two electron systems. For any molecular system composed of reasonably well defined electron pair bonds, such methods provide excellent starting points for inclusion of additional corrections such as those from three electron correlations. Such three electron correlations although computationally expensive, are crucial to reach quantitative

accuracy (Raghavachari and Anderson, 1996). In general, correlation techniques are computationally more expensive than HF methods. Many correlation techniques involve an iterative solution of a set of coupled equations.

5.4.2 Configuration interaction (CI)

Among the many schemes introduced to overcome the deficiencies of Hartree-Fock method, the most simple and general technique to address the correlation problem is *configuration interaction* (Shavitt, 1977; Schaefer, 1977; Boys, 1950). Configuration interaction is a straight forward application of the linear variational technique to the calculation of electronic wavefunctions. A linear combination of configurations (Slater determinants) is used to provide a better variational solution to the exact many-electron wavefunction.

The most important type of correlation effect which contributes to chemical bonding is usually termed as *left-right* correlation. For H_2, this refers to the tendency that when one electron is near the first hydrogen, the other electron tends to be near the second hydrogen. This is absent in the HF method where the spatial positions of the two electrons occupying the lowest bonding molecular orbital are uncorrelated. The problem gets worse as the two atoms move apart and dissociate. Qualitatively, this can be corrected by including a second configuration where both electrons occupy the anti-bonding orbital. While this is unfavorable energetically, a mixture of the HF configuration with this second configuration provides a better description of the system. This is referred to as “configuration interaction”. The second configuration has only a small weight at the equilibrium distance in H_2 but its weight increases as the bond

distance increases until the configurations have equal weights at dissociation. Such a left-right correlation is included in valence-bond-type wave functions.

The exact wave function Ψ can not be expressed as a single determinant. CI method constructs other determinants by replacing one or more occupied orbitals within the Hartree-Fock determinant with a virtual orbital. In a single substitution, a virtual orbital Φ_a replaces an occupied orbital Φ_i within the determinant. This is equivalent to exciting an electron to a higher energy orbital. Similarly in a double substitution, two occupied orbitals are replaced by virtual orbitals and triple substitution would exchange three orbitals and so on.

CI wave function mixes the Hartree-Fock wave function with single, double, triple, quadruple … excited configurations, and the coefficients which determine the amount of mixing are determined variationally. The full CI method forms the wave function Ψ as a linear combination of Hartree-Fock determinant and all possible substituted determinants:

$$\Psi = b_0 \Psi_0 + \sum_{s>0} b_s \Psi_s \,, \tag{5.17}$$

where the $_0$-indexed term is the Hartree-Fock level and s runs over all possible substitutions. The b's are the set of coefficients to be solved for by minimizing the energy of the resultant wave function. If all possible excited configurations are included, the method gives the exact solution within the space spanned by a given basis set. Full CI is the most complete non-relativistic treatment of the molecular system possible, with the limitation imposed by the chosen basis set.

5.4.3 Limited configuration interaction

The full CI method is size consistent and variational. However, it is very computationally expensive and impractical for large systems, since the number of configurations in full CI expansion grows exponentially with the size of the system. Practical CI methods augment the Hartree-Fock by adding only a limited set of substitutions. CIS method adds single excitation to Hartree-Fock determinant, while CISD adds singles and doubles. The CISD method (Pople *et al.*, 1977; Krishnan *et al.*, 1980; Raghavachari and Pople, 1981) is an iterative technique where the computational dependence of each iteration scales as the sixth power of the size of the system. A disadvantage of the limited CI methods is that they are not size consistent, i.e. the energy does not scale linearly with the size of the system, and CISD energy is not additive for infinitely separated systems. For example, the CISD energy for two infinitely separated He atoms is different from twice the energy of a single He atom.

Quadratic configuration interaction (QCI) technique (Pople *et al.*, 1987) introduces size consistency in CISD theory. The CISD method consists of a set of linear equations in the configuration expansion coefficients (for single and double excitations) which are solved iteratively. In the QCISD method (Gauss and Cremer, 1988; Trucks and Frisch, 1998; Salter *et al.*, 1989), these equations are modified by the introduction of additional terms, quadratic in the expansion coefficients, which make the method size consistent.

5.4.4 Møller-Plesset Perturbation theory (MP)

MP theory (Møller and Plesset, 1934) treats the electron correlation as a perturbation on the Hartree-Fock problem. It adds higher excitations to the Hartree-Fock

theory as a non-iterative correction. The wave function and energy are expanded in a power series of the perturbation. The Hamiltonian is expressed in two parts:

$$H = H_0 + H', \tag{5.18}$$

where H_0 is called the zeroth order Hamiltonian, and H' is the perturbation. H_0 is chosen such that the Schrödinger equation $H_0\Psi_0 = E_0\,\Psi_0$ can be solved exactly for the zeroth order eigenfunctions Ψ_0 and the corresponding zeroth order energy eigenvalues E_0. It is assumed that perturbation is sufficiently small that its presence does not appreciably alter the eigenfunctions of the system.

Hatree-Fock energy is correct to the first order. Hence the perturbation energies start contributing from second order. While performing MP calculations, first HF calculation is performed and then second, third, fourth, and fifth order perturbation corrections are computed. The final ground state energy is given by

$$E_{\text{ground state}} = E_{HF} + E^{(2)} + E^{(3)} + E^{(4)} + E^{(5)} + \dots, \tag{5.19}$$

where E_{HF} is the Hartree-Fock energy and $E^{(i)}$ are the successive Møller-Plesset perturbation corrections.

In perturbation theory, the different correlation contributions emerge through their interaction with the starting HF wave function Ψ_0. Since the Hamiltonian contains only one- and two-electron terms, only single and double excitations can contribute via direct coupling to Ψ_0 in the lowest orders. However, the self-consistent optimization of the HF wave function prevents direct mixing between single excitations and Ψ_0. Thus, the second and third-order energies have contributions only from double excitations. In higher orders, there is indirect coupling via double excitations, and thus the fourth and fifth-order energies have contributions from single, double, triple, and quadruple excitations.

A common notation denoted as MPn (Raghavachari and Pople, 1978; Raghavachari *et al.,* 1980; Frisch *et al.,* 1980) is used. Thus, MP2 (Head-Gordon *et al,* 1988; Frisch *et al*, 1990; Head-Gordon, 1994; Trucks *et al*, 1998; Saebo and Almlof, 1989), MP3 (Pople *et al*, 1977; 1976), MP4 (Raghavachari and Pople, 1978) denote the total energies correct to second, third, fourth … order respectively. Perturbation theory truncated at any order is size consistent. MP2, MP3, MP4, and MP5 methods scale as the fifth, sixth, seventh and eighth power of the size of the system. If triples are excluded from the MP4 method (Raghavachari and Pople, 1978; Raghavachari *et al.,* 1980; Frisch *et al.,* 1980; Bartlett and Shavitt, 1977; Bartlett and Purvis, 1978; Bartlett *et al.,* 1983; Wilson and Saunders, 1979) the resulting MP4 (SDQ) technique (Raghavachari and Pople, 1978) can be evaluated with sixth order computational dependence. Fifth-order theory (MP5) (Kucharski and Bartlett, 1986; Kucharski *et al.,* 1989; Bartlett *et al.,* 1990; Raghavachari *et al.,* 1990) has been implemented though feasible only for small systems. Analytical derivatives of the potentials energy can be computed for MP2 (Frisch *et al*, 1990; Pople *et al*, 1979; Handy and Schaefer, 1984), MP3, and MP4 (SDQ) methods (Trucks *et al*, 1988). Analytical frequencies can be computed for MP2 method (Head-Gordon, 1994; Trucks *et al,* 1998). MP2 is the most applicable method for electron correlation effects for large systems. It can be implemented without requiring the storage of two-electron integrals and many other intermediate quantities which makes it possible to study large systems such as C_{60}.

5.5 Density functional theory (DFT)

Shortly after the formulation of quantum mechanics in the mid 1920's, Thomas (1926) and Fermi (1928) introduced the idea of expressing the total energy of a system as

a functional of the total electron density. Because of the crude treatment of the kinetic energy term, i. e. the absence of molecular orbitals, the accuracy of these early attempts was far from satisfactory. It was not until the 1960's that an exact theoretical framework called density functional theory (DFT) was formulated by Hohenberg and Kohn (1964) and Kohn and Sham (1965) that provided the foundation for accurate calculations. Earlier, motivated by the search for practical electronic structure calculations.

In contrast to the Hartree-Fock picture, which deals with a description of individual electrons interacting with the nuclei and all other electrons in the system, in density functional theory, the total energy is decomposed into three contributions, a kinetic energy, a Coulomb energy due to classical electrostatic interactions among all charged particles in the system, and a term called the exchange-correlation energy that captures all many-body interactions.

In density functional theory (Hohenberg and Kohn, 1964; Kohn and Sham, 1965), correlation energy along with exchange energy is treated as a functional of the three dimensional electron density. DFT partitions the electronic energy as:

$$E = E^{T} + E^{V} + E^{J} + E^{XC}, \tag{5.20}$$

where E^{T} is the kinetic energy term, E^{V} includes terms describing the potential energy of the nuclear-electron attraction and of the repulsion between pairs of nuclei, E^{J} is the electron-electron repulsion term and E^{XC} is the exchange correlation term.

All terms, except the nuclear-nuclear repulsion in Eqn.(5.20) are functions of ρ the electron density. The term E^{XC} accounts for the exchange energy arising from the antisymmetry of the quantum mechanical wavefuncion, and dynamic correlation in the motions of the individual electrons. Hohenbrg and Kohn (1964) demonstrated that E^{XC} is

determined entirely by the electron density, i.e. it is a function of electron density. The term E^{XC} is usually approximated as an integral involving only the spin densities and possibly their gradients as given by

$$E^{XC}(\rho) = \int f(\rho)(\vec{r}), \rho_B(\vec{r}), \nabla\rho_\alpha(\vec{r}), \nabla\rho_\beta(\vec{r})\, d^3\vec{r}\,, \tag{5.21}$$

ρ_α and ρ_β refer to α and β spin density, and total electron density ρ is ($\rho_\alpha + \rho_\beta$).

E^{XC} is separated into exchange and correlation parts corresponding to same spin and mixed spin interactions as given by

$$E^{XC}(\rho) = E^X(\rho) + E^C(\rho), \tag{5.22}$$

the terms $E^X(\rho)$ and $E^C(\rho)$ are again functionals of electron density, termed as exchange functional and correlation functional. Local functionals depend on the electron density, while gradient corrected functionals depend on both electron density ρ and its gradient $\nabla\rho$.

In local density approximation (LDA) approximation, the exchange-correlation energy is taken from the known results of the many-electron interactions in an electron system of constant density (homogeneous electron gas). The LDA assumes that at each point in a molecule or solid there exists a well defined electron density; an electron at such a point experiences the same many-body response by the surrounding electrons as if the density of these surrounding electrons had the same value throughout the entire space as at the point of the reference electron. The exchange-correlation energy of the total molecule or solid is then the integral over the contributions from each volume element. The contributions are different from each volume element depending on the local electron density. The local exchange functional is defined as

$$E_{LDA}^{X} = -\frac{3}{2}\left(\frac{3}{4\pi}\right)^{1/3} \int \rho^{4/3}\, d^3\vec{r}\,. \tag{5.23}$$

In DFT, the total electron density is decomposed into one-electron densities, which are constructed from one-electron wave functions. These one-electron wave functions are similar to those of Hartree-Fock theory. For molecular systems, DFT leads to a molecular orbital (MO) picture in analogy to the Hartree-Fock approach.

DFT has been successfully extended to open-shell systems and magnetic solids (von Barth and Hedin, 1972; Gunnarsson *et al.,* 1972). In these cases, the local exchange-correlation energy depends not only on the local electron density, but also on the local spin density (which is the difference between the electron density of spin-up electrons and that of spin-down electrons). The resulting generalization of LDA is called local spin density approximation (LSDA).

Within the local density approximation, binding energies are typically overestimated. Any real system is spatially inhomogeneous, i.e. it has spatially varying density. The accuracy could be improved if the rate of this variation is included in the functional. Gradient expansion approximation (GEA) tries to systematically calculate the gradient corrections to the LDA. In generalized gradient approximations (GGA), instead of power series gradient expansions, generalized functions are used.

Inclusion of non-local gradient corrections improves the values of binding energies. Becke (1988) formulated the gradient corrected exchange functional based on the LDA exchange functional given as

$$E_{Becke}^{X} = E_{LDA}^{X} - \gamma \int \frac{\rho^{4/3}\, x^2}{(1 + 6\gamma \sinh^{-1}(x))}\, d^3\vec{r}\,, \tag{5.24}$$

where $x = \rho^{-4/3}\left|\nabla\rho\right|,^{\gamma}$ is a parameter chosen to fit known exchange energies of the inert atoms. Becke functional is defined as a correction to the local LDA exchange functional. Self consistent Kohn- Sham DFT calculations are performed in an iterative manner analogous to the SCF computation. Becke formulated which include Hartree- Fock and DFT exchange along with DFT correlation as;

$$E_{hybrid}^{XC} = c_{HF} E_{HF}^{X} + c_{DFT} E_{DFT}^{XC}. \tag{5.25}$$

For example Becke style three parameter functional denoted as B3LYP (Becke, 1993) is defined as:

$$E_{B3LYP}^{XC} = E_{LDA}^{X} + c_0 (E_{HF}^{X} - E_{LDA}^{X}) + c_X \Delta E_{Becke}^{X} + E_{VWN3}^{C} + c_C (E_{LYP}^{C} - E_{VWN3}^{C}). \tag{5.26}$$

The parameter c_0 allows any combination of Hartree- Fock and LDA local exchange to be used. The local correlation functional *VWN3* is corrected by the *LYP* correlation correction.

Table 5.1 lists various model chemistries definable via *ab initio* methods and standard basis sets. Each cell in the chart defines a model chemistry.

Table 5.1 Effect of various model chemistries and basis sets on accuracy of the solution to Schrödinger's equation (Foresman and Frisch).

Basis Set Type	Electron Correlation → HF	MP2	MP3	MP4	QCISD(T)	...	Full CI
Minimal						...	
Split-valence						...	
Polarized						...	
Diffuse						...	
High Ang Moment						...	
...	...	...	...	...	...	...	
∞	HF Limit					...	Schroedinger Equation

The columns correspond to different theoretical methods and the rows to different basis sets. The level of correlation increases from left to right across any row, with Hartree-Fock method at the extreme left (including no correlation), and the full configuration interaction method at the right (which fully accounts for electron correlation). The rows of the chart correspond to increasingly larger basis sets. The bottom row of the chart represents a completely flexible basis set. The cell in the lower right corner of the chart represents the exact solution of the Schrödinger equation.

CHAPTER 6

Neural Networks

As the term *neural network* (NN) implies, this approach is aimed towards modeling real networks of neurons in the brain. It was inspired by a number of features of the brain that would be desirable in artificial systems, such as its robustness and fault tolerance, its flexibility and the ability to deal with fuzzy and noisy information and its highly parallel structure.

6.1 Biological neurons

The brain is composed of about 10^{11} highly connected elements called neurons, each of which is connected to about 10^{4} other neurons. Fig 6.1 shows the schematic of biological neuron.

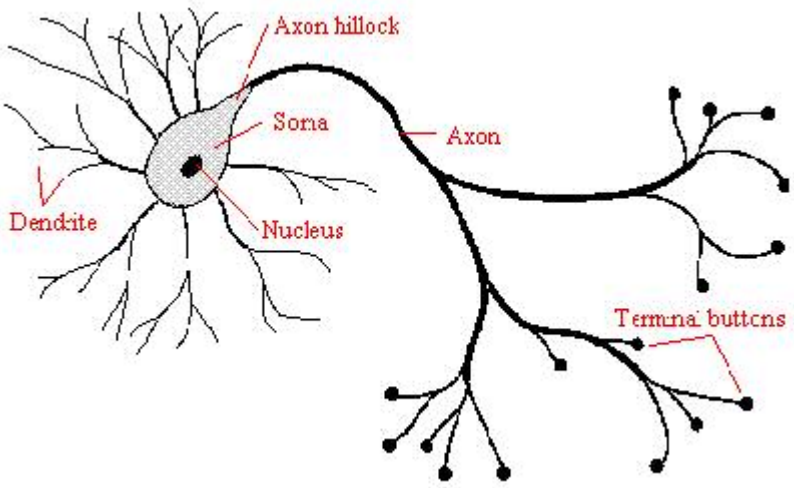


Fig 6.1 Schematic of biological neuron

A neuron's dendritic tree is connected to thousands of neighboring neurons. Each neuron receives electrochemical inputs from other neurons at the dendrites. When one of those neurons fire, a positive or negative charge is received by one of the dendrites. The strengths of all the received charges are added together through the processes of spatial and temporal summation. Spatial summation occurs when several weak signals are converted into a single large one while temporal summation converts a rapid series of weak pulses from one source into one large signal. The aggregate input is then passed to the soma (cell body). The soma and the enclosed nucleus do not play a significant role in the processing of incoming and outgoing data. Their primary function is to perform the continuous maintenance required to keep the neuron functional. If the sum of these electrical inputs is sufficiently powerful to activate the neuron, it transmits an electrochemical signal along the axon, and passes this signal to the other neurons whose dendrites are attached at any of the axon terminals. These attached neurons may then fire. The part of the soma that does concern itself with the signal is the axon hillock. If the aggregate input is greater than the axon hillock's threshold value, then the neuron fires, and an output signal is transmitted down the axon. The strength of the output is constant, regardless of whether the input was just above the threshold, or a hundred times as great. The output strength is unaffected by the many divisions in the axon; it reaches each terminal button with the same intensity it had at the axon hillock. This uniformity is critical in an analogue device such as a brain where small errors can be critical, and where error correction is more difficult than in a digital system. It is important to note that a neuron fires only if the total signal received at the cell body exceeds a certain level. The neuron either fires or it does not, there are not different grades of firing.

Each terminal button is connected to other neurons across a small gap called a synapse, as shown in Fig 6.2 . The physical and neuro-chemical characteristics of each synapse determine the strength and polarity of the new input signal. This is where the brain is most flexible.

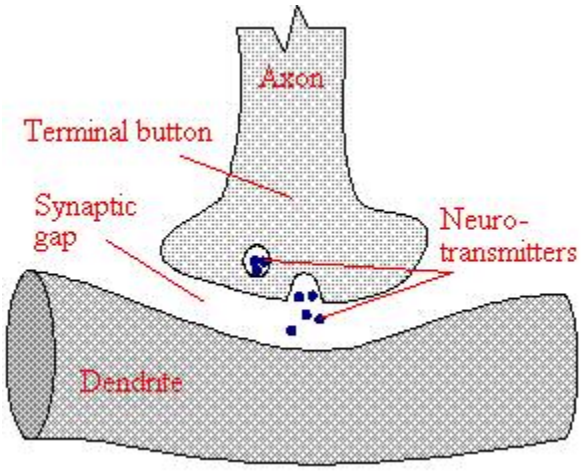


Fig 6.2 Synaptic gap

Changing the constitution of various neuro-transmitter chemicals can increase or decrease the amount of stimulation that the firing axon imparts on the neighboring dendrite. Altering the neurotransmitters can also change whether the stimulation is excitatory or inhibitory. Many drugs, such as alcohol have dramatic effects on the production or destruction of these critical chemicals. The infamous nerve gas, sarin, can kill because it neutralizes a chemical (acetylcholinesterase) that is normally responsible for the destruction of a neurotransmitter (acetylcholine). This means that once a neuron fires, it keeps on triggering all the neurons in the vicinity and one no longer has control over muscles.

So, from a very large number of extremely simple processing units (each performing a weighted sum of its inputs, and then firing a binary signal, if the total input

exceeds a certain level) the brain manages to perform extremely complex tasks. This is the model on which artificial neural networks (NN) are based. Thus far, artificial neural networks have not even come close to modeling the complexity of the brain. It is worth noting that even though biological neurons are slow compared to electrical circuits, the brain is able to perform many tasks much faster than any conventional computer because of the massively parallel structure of biological neural networks.

While in actual neurons the dendrite receives electrical signals from the axons of other neurons, in an artificial neuron (perceptron) these electrical signals are represented as numerical values. At the synapses between the dendrite and axons, electrical signals are modulated in various amounts. This is also modeled in an artificial neuron by multiplying each input value by a value called the weight. An actual neuron fires an output signal only when the total strength of the input signals exceeds a certain threshold. We model this phenomenon in a ANN by calculating the weighted sum of the inputs to represent the total strength of the input signals, and applying a step function on the sum to determine its output. Fig 6.3 shows a simple model of a neuron.

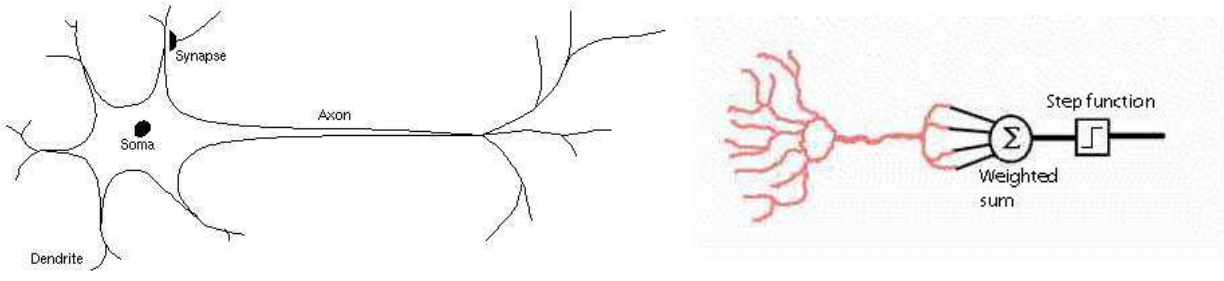


Fig 6.3 Model of a neuron

6.2 History of multilayer neural networks

The first step toward artificial neural networks came in 1943 when Warren McCulloch and Walter Pitts (1943) wrote a paper “A logical calculus of the ideas immanent in nervous activity” describing how neurons might work that united the studies of neurophysiology and mathematical logic. They modeled a simple neural network with electrical circuits and showed the network in principle, could compute any arithmetic function. Training of the neural network was introduced when Hebb (1949) in 1949 published the book "The organization of behavior," which pointed out the fact that neural pathways are strengthened each time they are used, a concept fundamentally essential to the ways in which humans learn. If two nerves fire at the same time, the connection between them is enhanced. Frank Rosenblatt (1958) invented the perceptron and associated learning rule. It was built in hardware. A single-layer perceptron was found to be useful in classifying a continuous-valued set of inputs into one of two classes. This was the first practical application of artificial neural networks. In 1959, Bernard Widrow and Marcian Hoff developed models called "ADALINE" (ADAptive LInear Neuron) and "MADALINE” (Multiple ADAptive LInear Neuron). ADALINE was developed to recognize binary patterns so that if it was reading streaming bits from a phone line, it could predict the next bit. MADALINE was the first neural network applied to a real world problem, using an adaptive filter to eliminate echo on phone lines. In 1962, Widrow and Hoff developed a learning rule to train ADALINE networks. Minsky and Papert (1969) showed that perceptron and ADALINE networks were limited for classification problems in which the two classes were linearly separable. These networks were incapable of handling nonlinear classification problems, such as exclusive-or (*XOR*)

problem. Rosenblatt and Widrow were aware of these limitations and proposed multilayer networks. First description of an algorithm to train multilayer networks was given by Webros (1974). He described it for general networks with neural networks as a special case. It was rediscovered by Rumelhart, Hinton, and Williams (1986), Parker (1985) and Le Cun (1985). Backpropagation was developed by generalizing Widow-Hoff learning rule to multilayer neural networks with nonlinear differentiable transfer functions.

6.3 Neural networks versus conventional computers

Neural networks take a different approach to problem solving than that of conventional computers. Conventional computers use an algorithmic approach i.e. the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the computer needs to follow are known the computer cannot solve the problem. That restricts the problem solving capability of conventional computers to problems that we already understand and know how to solve particularly for repetitive tasks. But computers would be so much more useful if they could do things that we do not exactly know how to do. Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve a specific problem. Neural networks learn by example. They are not programmed to perform a specific task. The examples must be selected carefully otherwise the network might function incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable.

On the other hand, conventional computers use a predictive approach to problem solving. The way the problem is to be solved must be known and stated in unambiguous instructions. These instructions are then converted to a high level language program and then into machine code that the computer can understand. Their operation is predictable.

Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks that are more suited to an algorithmic approach, such as arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency. Neural networks have shown to be good at problems which are easy for a human but difficult for a traditional computer, such as image and voice recognition and predictions based on past knowledge.

6.4 Model of a neuron

Fig 6.4 shows a single input neuron, which forms the building block for the Artificial Neural Network (ANN). The scalar input *p* is multiplied by the scalar weight *w*, the other input 1 is multiplied by the bias *b*. the summer adds *wp* and *b* to form the net input *n*, which goes into a transfer function *f* to produce a scalar output *a*.

w and *b* are adjustable scalar parameters of the neurons, which are adjusted by the learning rule so as to minimize the errors (i.e. neuron outputs and the target values).

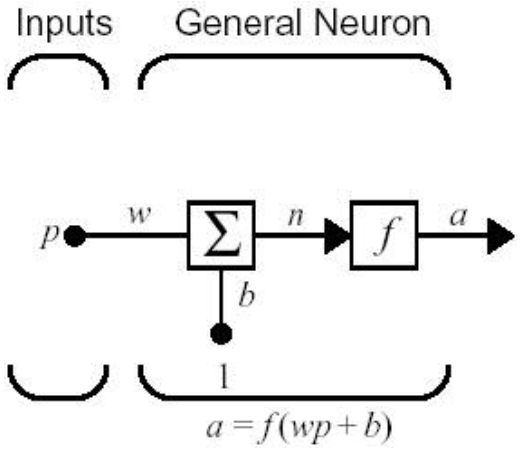


Fig 6.4 Single input neuron (Hagan *et al.* 1996)

6.5 Multilayer network

Fig 6.5 shows a neural network with multiple layers with each layer having multiple neurons. Each layer has its own weight matrix *W* and bias vector *b*. The superscript to each parameter indicates the layer it is associated with. The network has *R* inputs. The outputs from one layer are inputs to the next layer.

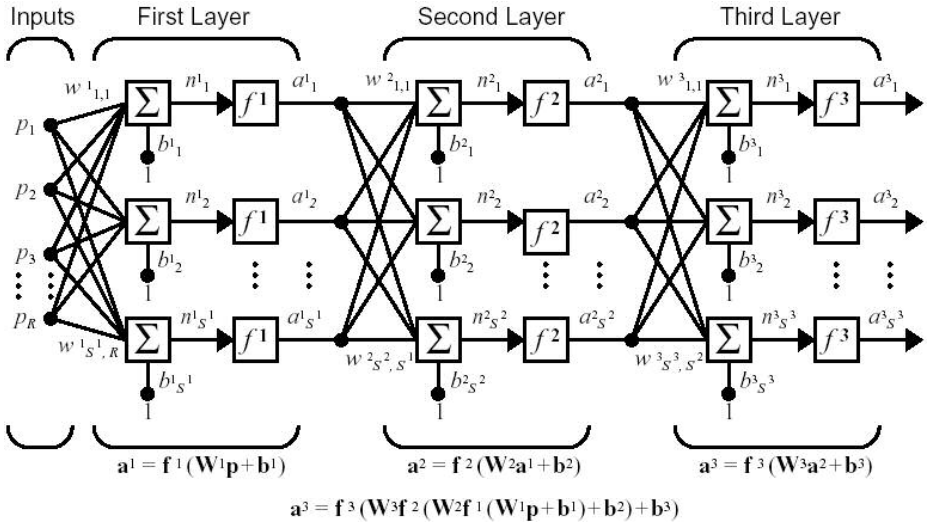


Fig 6.5 Multilayer network (Hagan *et al.* 1996)

A layer whose output is the output of the entire neural network is called as the output vector, whereas other layers are called as hidden layers. The number of neurons in the

input layer is the same as the number of input variables and the number of neurons in the output layer is same as the number of outputs. The number of neurons to be used for the hidden layer is decided by the designer depending upon the complexity of the function involved.

6.6 Transfer functions

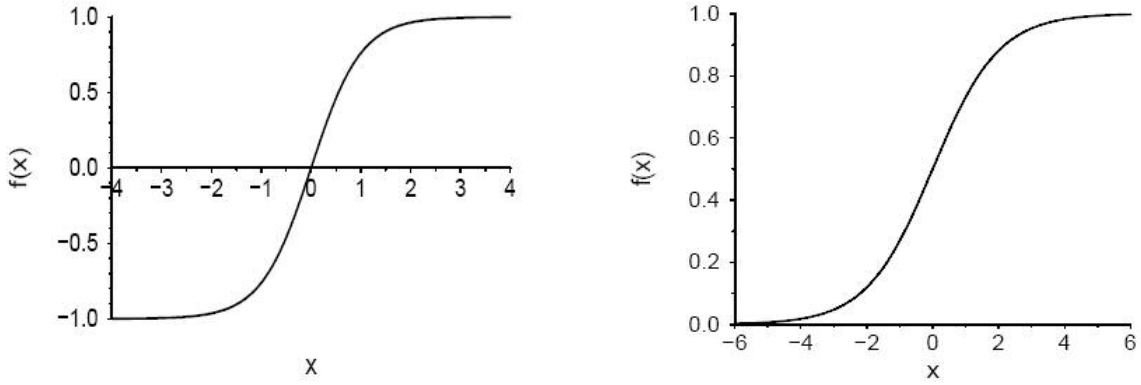
A transfer function is a linear or non-linear function of the net input to the neuron. Nonlinear transfer functions give neural networks their nonlinear capabilities. The function must be differentiable for the optimization of the parameters and is desirable to saturate at both extremes. The most common forms of transfer functions are the monotonically increasing sigmoidal or hyperbolic tangent function.

Hyperbolic tangent function

$$f(x)=\frac{e^{n}-e^{-n}}{e^{n}+e^{-n}}. \tag{6.1}$$

Log sigmoid function

$$f(x)=\frac{1}{1+e^{-n}}. \tag{6.2}$$



(a) (b)

Fig 6.6 a) Hyperbolic tangent function ranging from [-1, +1] for x = $\pm\infty$

b) Log sigmoid function ranging from [-1,+1] for x = $\pm\infty$

These functions are also called as squashing functions because of their asymptotic behavior at $\pm\infty$. Convergence with functions which are symmetric about the origin, like the hyperbolic tangent is faster (LeCun, 1998)

6.7 Neural network training

Neural network training is a procedure for modifying weight matrices and bias vectors of the network to meet certain goals, such as minimizing the difference between the network output and target values. It is equivalent to performing the non-linear optimization of the network parameters. There are two approaches for neural network training: supervised learning and unsupervised learning.

In supervised training, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights which control the network. This process occurs over and over and the weights are continually adjusted. During the training of a network the same set of data is processed many times as the weights and biases are adjusted. It is very important to have enough data points for the training so as to cover the configuration space adequately. The number of data points depends on the number of input variables, as the number of input variables increase. It increases the dimensionality of the problem and more points are required in order to cover the entire configuration space. The number of data points required also depends on the complexity of the problem. Also, it is equally important to choose the input variables carefully. Sometimes it may not be obvious as to which variables to choose so that the input data contains complete information from which the target output is derived. Supervised learning has applications in function approximation

and pattern classification problems where the number and type of patterns is known beforehand.

In unsupervised training, the network is provided with the inputs but not with the desired outputs. The system itself decides what features to use to group the input data. This is referred to as self-organization. These networks look for regularities or trends in the input to adapt the network parameters. Unsupervised learning has applications in clustering operation. They learn to categorize the input patterns into a finite number of classes.

6.7.1 Supervised training and parameter optimization

The supervised training is done by comparing the output with known target values. The optimization of network parameters is performed by some iterative optimization scheme until the desirable solution by the scalar cost function is reached. Normally the cost function is taken as the sum of the squared difference between the network output and the target values.

$$E(e^2) = E\left[(t-a)^2\right], \tag{6.3}$$

where t is the target vector, a is the vector of neural network output and e is the difference between the target and the neural network output. $E(e^2)$ is the expectation of the sum squared error. The training process can be classified into two types: incremental training and batch training. In incremental training the weights and biases of the network are updated each time an input is presented to the network, while in batch training, the weights and biases are updated after all inputs are presented. The incremental training is comparatively faster since in batch training the redundancy or the presence of patterns which are very similar leads to slower convergence. In incremental training the noise

present in the updates coming from the numerical calculation of the gradient after each successive sample can result in weights which are able to find a different minimum before getting stuck in a local minimum, unlike in batch training where the random initialization of the weights is very crucial. However such noise can prevent full convergence to the minimum. For batch training the conditions of convergence are well understood. Many optimization schemes, such as conjugate gradient method are applicable only for batch training. Also, the analysis of convergence rates is simpler.

In order to minimize the cost function, the training algorithm cycles through the following steps:

1. The network is presented with one (for incremental training) or all (for batch training) training samples consisting of both inputs and targets.
2. The network output is measured and the squared error between the target and the output is calculated.
3. The weights and biases are adjusted so as to minimize the cost function i.e. the squared error.
4. The procedure is repeated until the squared error reaches the desired limit.

The optimization of the parameters is usually done using gradient based methods, such as steepest descent or conjugate gradient or quasi Newton algorithms.

6.8 LMS (Least Mean Squared Error) algorithm

The LMS or Widrow–Hoff algorithm is based on approximate steepest descent algorithm. Widrow and Hoff noticed that the estimate of the mean squared error could be obtained by using squared error at each iteration.

$$\nabla F(x) = \nabla e^2(k), \tag{6.4}$$

where k is the iteration number.

The partial derivatives of squared error with respect to weights and biases at k^{th} iteration are given by:

$$\frac{\partial e^2(k)}{\partial w_{1,j}} = 2e(k)\frac{\partial e(k)}{\partial w_{1,j}}, \tag{6.5}$$

$$\text{and } \frac{\partial e^2(k)}{\partial b} = 2e(k)\frac{\partial e(k)}{\partial b}.$$

where R is the number of input variables.

$$\frac{\partial e(k)}{\partial w_{1,j}} = \frac{\partial [t(k) - a(k)]}{\partial w_{1,j}} = \frac{\partial}{\partial w_{1,j}}\left[t(k) - \left(\sum_{i=1}^{R} w_{1,i}\, p_i(k) + b \right) \right] = -p_j(k), \tag{6.6}$$

where $p_i(k)$ is the i^{th} element of the input vector at the k^{th} iteration.

Similarly,
$$\frac{\partial e(k)}{\partial b} = -1. \tag{6.7}$$

Therefore $\nabla F(x) = \nabla e^2(k) = -2e(k)\begin{bmatrix} p \\ 1 \end{bmatrix}$ … where p is the input vector and 1 is constant input for bias. That is, the approximate gradient is given by multiplying the input with the error. Now the steepest descent method produces proceeds as

$$x_{k+1} = x_k - \alpha \nabla F(x)_{|x = xk}. \tag{6.8}$$

LMS algorithm can be written in matrix notation as:

$$W_{(k+1)} = W_{(k)} + 2\alpha\, e(k)\, p^T_{(k)},$$

$$\text{and } b_{(k+1)} = b(k) + 2\alpha\, e(k). \tag{6.9}$$

6.9 Backpropagation algorithm

Neural networks employing only one layer of neurons can solve only linearly separable classification problems. First description of an algorithm to train multilayer networks was given by Webros (1974). He described it for general networks with neural networks as a special case. It was rediscovered by Rumelhart (1986), Parker (1985) and Le Cun (1985). Backpropagation was developed by generalizing Widow-Hoff learning rule to multilayer neural networks with nonlinear differentiable transfer functions. It is also a gradient descent algorithm where the parameters are adjusted along the negative of the gradient of the cost function. The term *backpropagation* refers to the manner in which the gradient is computed for nonlinear multilayer networks. With single layer network, the weights are adjusted based on the error values as in LMS (Widrow-Hoff) learning. However, with multilayer networks, it is less obvious how to calculate the error for the hidden layers as the error from the output layer is not an explicit function of weights in the hidden layers.

Chain rule is applied to compute the gradient for the steepest descent algorithm.

$$\frac{\partial F}{\partial w_{i,j}^m} = \frac{\partial F}{\partial n_i^m} \cdot \frac{\partial n_i^m}{\partial w_{i,j}^m}, \tag{6.10}$$

$$\text{and } \frac{\partial F}{\partial b_i^m} = \frac{\partial F}{\partial n_i^m} \cdot \frac{\partial n_i^m}{\partial b_i^m}.$$

where n_i^m is the net input to the i^{th} neuron in the m^{th} layer, $w_{i,j}^m$ and b_i^m are the weights and biases the layer *m*.

Now the net input n_i^m is computed as:

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m \, a_j^{m-1} + b_i^m , \tag{6.11}$$

$$\text{and } \frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1} , \frac{\partial n_i^m}{\partial b_i^m} = 1.$$

Now, a term *sensitivity* is defined as the change in the cost function F with respect to change in the i^{th} element of the net input at the layer m.

$$s_i^m = \frac{\partial F}{\partial n_i^m} . \tag{6.12}$$

Therefore

$$\frac{\partial F}{\partial w_{i,j}^m} = s_i^m \, a_j^{m-1} , \tag{6.13}$$

$$\text{and } \frac{\partial F}{\partial b_i^m} = s_i^m .$$

Now, the steepest algorithm can be expressed as:

$$W^m (k+1) = W^m (k) - \alpha \, s^m (a^{m-1})^T , \tag{6.14}$$
$$\text{and } b^m (k+1) = b^m (k) - \alpha \, s^m .$$

where W^m and b^m are weight matrix and bias vector at the layer m.

$$s^m = \frac{\partial F}{\partial n^m} = \begin{bmatrix} \partial F / \partial n_1^m \\ \partial F / \partial n_2^m \\ . \\ . \\ . \\ \partial F / \partial n_{s^m}^m \end{bmatrix} \tag{6.15}$$

Next step is to compute the sensitivities. This process involves a recurrence relationship in which the sensitivity at layer m is computed from the sensitivity at layer $m+1$, which gives it the name *backpropagation.* Now the Jacobian matrix is defined as:

$$\frac{\partial n^{m+1}}{\partial n^m} = \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \frac{\partial n_1^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_1^{m+1}}{\partial n_{S^m}^m} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} & \frac{\partial n_2^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_2^{m+1}}{\partial n_{S^m}^m} \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_1^m} & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_{S^m}^m} \end{bmatrix} \tag{6.16}$$

Now $\frac{\partial n_i^{m+1}}{\partial n_j^m}$ is computed as:

$$\frac{\partial n_i^{m+1}}{\partial n_j^m} = \frac{\partial\left(\sum_{l=1}^{S^m} w_{i,l}^{m+1} a_l^m + b_i^{m+1}\right)}{\partial n_j^m}, \tag{6.17}$$

$$= w_{i,j}^{m+1} \frac{\partial a_j^m}{\partial n_j^m} = w_{i,j}^{m+1} \frac{\partial f^m(n_j^m)}{\partial n_j^m}.$$

Hence, the Jacobian matrix can be written as:

$$\frac{\partial n^{m+1}}{\partial n^m} = W^{m+1} \dot{F}^m(n^m), \tag{6.18}$$

$$\dot{F}^m(n^m) = \begin{bmatrix} \dot{f}(n_1^m) & 0 & \ldots & 0 \\ 0 & \dot{f}(n_2^m) & \ldots & 0 \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ 0 & 0 & & \dot{f}(n_{S^m}^m) \end{bmatrix}$$

And the recurrence relation can be written as:

$$s^m = \dot{F}^m(n^m)(W^{m+1})^T s^{m+1}, \tag{6.19}$$

Thus, the sensitivities are propagated backward through the network from last to first layer, which gives it the name *backpropagation.*

For the output layer the sensitivities are computed as:

$$s = -2\dot{F}(n)(t-a), \tag{6.20}$$

where t is the target value and a is the neural network output.

6.10 Numerical optimization techniques

The mean squared error of a single layer network with linear transfer function is a quadratic function with only one minimum and a constant curvature. In such cases, the LMS algorithm guarantees the convergence, so long as the learning rate is not too large. But for a multilayer network the mean squared error may be a non-quadratic function where the curvature varies drastically over the parameter space which makes it difficult to choose the learning rate. To overcome some of these difficulties heuristic techniques, such as variable learning rate, using momentum and rescaling the variables are employed. Another approach employs standard numerical optimization techniques, such as conjugate gradient and quasi Newton method.

6.10.1 Conjugate gradient methods

The backpropagation algorithm adjusts the weights along the steepest descent direction. Although this is the direction in which the function decreases most rapidly, this may not produce faster convergence. Instead of using orthogonal search directions as in the steepest descent method, conjugate gradient method adjusts the search direction so as to pass through the minimum of the function. The conjugate gradient method uses the information on the first derivative of the error function only and also has a quadratic convergence property i.e. it converges to the minimum of a quadratic function in a finite number of steps. This is a *O(N)* method.

6.10.2 Newton and quasi-Newton methods

Newton method

The cost function i.e. the mean squared error can be expanded using Taylor's series as:

$$F(w+\Delta w) = F(w) + g^T \cdot \Delta w + \frac{1}{2}\Delta w^T \cdot H \cdot \Delta w + \ldots, \tag{6.21}$$

where g is the gradient vector and H is the Hessian matrix defined as:

$$g = \frac{\partial F(w)}{\partial w}, \tag{6.22}$$

$$H = \frac{\partial^2 F(w)}{\partial w^2}$$

Quadratic approximation of the error surface can be obtained by ignoring higher order terms in the Taylor's series expansions. The optimum value of Δw i.e. the adjustments to the weights is:

$$\Delta w = -H^{-1} \cdot g\ . \tag{6.23}$$

Newton method requires computing the Hessian matrix which involving second order derivatives and then finding its inverse which takes $O(N^3)$ iterations. But this does not guarantee the convergence. If the error surface is not quadratic, then the Hessian matrix may not be always positive definite and the algorithm diverges.

In quasi Newton method, Hessian matrix is estimated by some positive definite matrix, which ensures the convergence. The Hessian matrix is further approximated as:

$$H \cong 2\,J^T\,J\ , \tag{6.24}$$

where J is the Jacobian matrix.

6.10.3 Levenberg-Marquardt algorithm

The Hessian matrix in Gauss-Newton method may not always invertible. Levenberg-Marquardt algorithm uses an approximation to the Hessian matrix as:

$$\Delta w = \left[J^T J + \mu I \right]^{-1} J^T, \tag{6.25}$$

where J is the Jacobian matrix consisting of the first derivatives of network errors with respect to weights and biases. When μ is zero, then it is similar to Gauss-Newton method, and when μ is large, it becomes the gradient descent algorithm. The Jacobian matrix can be computed through standard backpropagation technique that is less complex than computing the Hessian matrix (Hagan and Menhaj, 1994).

6.11 Bias/variance dilemma

One of the most serious problems that arise in connectionist learning by neural networks is *overfitting* of the provided training examples. This means that the trained function fits very closely the training data. However it does not generalize well, i.e. it can not model unseen data sufficiently well. The problem is to choose a function that both fits the training data as well as generalizes well over a specified range. If the function is parameterized so as to fit the training data perfectly using a higher order polynomial (which is equivalent to having more number of neurons in the hidden layers) then the fits would be drastically different for different sets of data randomly drawn from the same underlying function. More parameter flexibility has a benefit of fitting anything but at the cost of sensitivity to the variability in the data set. There is a variation between the average fit over all data sets and the fit for a single data set i.e. a variance introduced by the fits over multiple training sets. This can be addressed to some extent by having more number of data points for the fitting. There is a drawback to the flexibility afforded by

extra degrees of freedom in fitting the data that the amount of data required grows exponentially with the order of the fit. On the other hand, if very few parameters are available then the fit would be too restrictive and although the fit would not be as sensitive to the variability in the training data there would be a fixed error or bias no matter how much data is collected. This problem is referred to as *bias/variance dilemma* (Geman *et. al,* 1992). This becomes a problem when there are relatively few data points especially in high dimensional space.

Statistical bias is the complexity restriction that the neural network architecture imposes on the degree of fitting accurately to the target function. The statistical bias accounts only for the degree of fitting the given training data, but not for the level of generalization. Statistical variance is the deviation of the neural network training efficacy from one data sample to another that could be described by the same target function model. This statistical variance accounts for the generalization of whether or not the neural network fits the examples without regard to the specificities of the provided data. One approach to avoid overfitting is to deliberately add some random noise with a mean value of zero to the model, which makes it difficult for the network to fit any specific data point too closely. Another way of introducing noise is to use increment training, i.e. updating the weights after every data point is presented, and to randomly reorder the data points at the end of each training cycle. In this manner, each weight update is based on a noisy estimate of the true gradient.

6.11.1 Neural network growing and neural network pruning

To reduce the statistical bias neural network growing is employed, where initially the training is started with less number of neurons. During the training process when the

decrease in error falls below a certain threshold value, then more neurons are added. With a view to reduce the statistical variance, neural network pruning is employed where a large network is first trained and the relative importance of the weights is assessed. Less important weights are then removed without affecting the performance much to get a smaller network

6.12 Regularization and early stopping

Regularization involves modifying the performance function which is the mean squared error. The risk of overfitting can be reduced if the variance is used in the error function to penalize the neural network model with high curvature. The usual performance function is defined as:

$$F = Mean\,Squared\ Error\,(MSE) = \frac{1}{N}\sum_{i=1}^{N}\left(t_i - a_i\right)^2, \tag{6.26}$$

It is modified as:

$$F = \frac{1}{N}\sum_{i=1}^{N}\left(t_i - a_i\right)^2 + \gamma\, R(W), \tag{6.27}$$

where *R(W)* is the function of network parameters. Regularization is performed by weight decay (Hertz *et. al*, 1996). A common form of *R(W)* used to penalize the model is taken as the sum square of weights, since overfitting with large curvature occurs when the weights of the network become too large.

$$F = mse\,\mathrm{Re}\,gularization = \gamma \cdot mse + (1-\gamma)\,msw, \tag{6.28}$$

$$msw = \frac{1}{n}\sum_{j=1}^{n} w_j^2,$$

where γ is the performance ratio. Using this performance function will cause the network to have smaller weights and biases and this will cause the network response to be smoother and less likely to overfit.

Bayesian framework (MacKay, 1992) provides an approach to determine the optimal regularization parameter in an automated fashion. In this framework the weights and biases are assumed to be random variables with a specified distribution. The regularization parameters are related to unknown variances associated with these distributions. Bayesian regularization is implemented along with Levenberg- Marquardt algorithm (Foresee and Hagan, 1997).

6.13 Early stopping

In order to achieve good generalization, overfitting needs to be controlled. This can be achieved by early stopping. In this technique, the available data set is divided into three subsets, viz. training set, validation set, and testing set. Training set is used to compute the gradients and updating the weights and biases. Validation and testing sets are used to monitor the error. Generally, the error on the training and validation sets decrease during the initial phase of training. However, when the network starts overfitting, the error on the training set continues to decrease while that on the validation set starts increasing. When the error on the validation set increases for a specified number of iterations, the training is stopped. Fig 6.7 shows a typical variation of the error on the training and validation sets during neural network training.

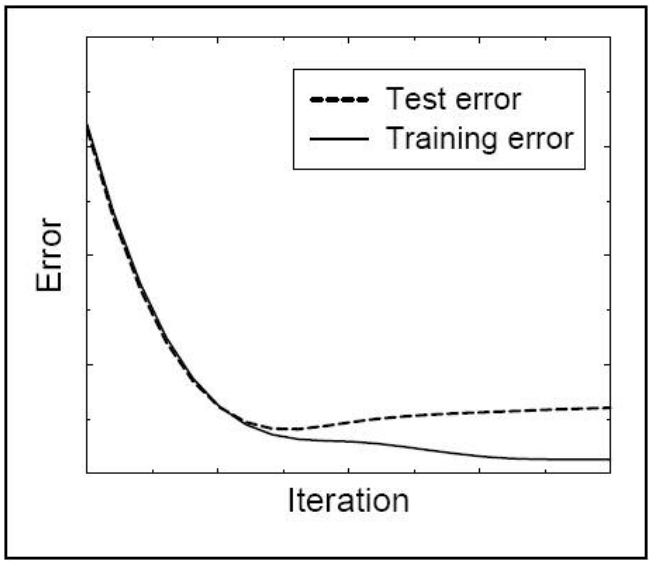


Fig 6.7 Evaluation of error during training

6.14 Normalizing the data

If the range of the output variables is quite large, then before training it is often useful to scale the inputs and targets so that they fall within a specified range. Since the range for nonlinear transfer functions, such as logsigmoid and hyperbolic tangent function is [0, 1] and [-1, +1], the inputs and targets are scaled within a range from [-1, +1] with the average close to zero.

$$p_i = 2 \cdot \frac{(p_i - \min p)}{(\max p - \min p)} - 1, \tag{6.29}$$

where *min p* and *max p* are the minimum and maximum of the input or target values.

6.15 Neural network derivatives

In general, for function approximation problems, the neural network is used to map the functional relationship between input and output variables. Once trained, this neural network is presented with a input vector to calculate the corresponding output. But in some cases the derivatives of the output variables with respect to the input variables

may be desired. An example of such an application would be molecular dynamics simulation, where a multilayer neural network is trained for the energy of a molecular cluster with bond distances and angles that define the geometry of the cluster as the input. During the molecular dynamics simulation. The force i.e. the first derivative of the energy with respect to coordinates of atoms is required.

6.15.1 First derivative with respect to weights

For the multilayer network, the network output is given as:

$$a^{m+1} = f^{m+1}(W^{m+1}a^{m} + b^{m+1}) \;\; for\, m = 0,1,...,M-1, \tag{6.30}$$

where the superscript represents the layer number. M is the number of layers of the network and input to the first layer is $a^0 = p$

Now,

$$\frac{\partial F}{\partial w_{i,j}^{m}} = \frac{\partial F}{\partial n_i^{m}} \cdot \frac{\partial n_i^{m}}{\partial w_{i,j}^{m}}, \tag{6.31}$$

$$\text{and } \frac{\partial F}{\partial b_i^{m}} = \frac{\partial F}{\partial n_i^{m}} \cdot \frac{\partial n_i^{m}}{\partial b_i^{m}}.$$

$$\frac{\partial F}{\partial w_{i,j}^{m}} = s_i^{m}\, a_j^{m-1}, \tag{6.32}$$

$$\text{and } \frac{\partial F}{\partial b_i^{m}} = s_i^{m}.$$

where the sensitivity is defined as $s_i^{m} = \frac{\partial F}{\partial n_i^{m}}$

6.15.2 Derivatives of the transfer function

The derivatives of the transfer functions are required to calculate the derivatives of the network output with respect to the network input.

Linear function:

$$f(x) = x, \tag{6.33}$$

$$\frac{df(x)}{dx} = 1,$$

$$\frac{d^2 f(x)}{dx^2} = 0.$$

Log sigmoid function

The log sigmoid transfer function is defined as:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{6.34}$$

$$\frac{df(x)}{dx} = e^{-x}(1 + e^{-x})^{-2} = f(x)\left(1 - f(x)\right),$$

$$\frac{d^2 f(x)}{dx^2} = f(x)\left(1 - f(x)\right)\left(1 - 2f(x)\right).$$

Hyperbolic tangent (tan sigmoid) function

The hyperbolic tangent function is defined as:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{6.35}$$

$$\frac{df(x)}{dx} = (1 - f^2(x)),$$

$$\frac{d^2 f(x)}{dx^2} = -2 f(x)(1 - f^2(x)).$$

6.15.3 First derivative with respect to inputs

For the multilayer network, the network output is given as:

$$a^{m+1} = f^{m+1}(W^{m+1}a^{m} + b^{m+1}) \;\; for\, m = 0,1,...,M-1, \tag{6.36}$$

where the superscript represents the layer number. M is the number of layers of the network and the input to the first layer is $a^0 = p$

The sensitivity is defined as:

$$s^{M} = \frac{da^{M}}{dn^{M}} = \dot{F}^{M}(n^{m}) \cdot W^{M+1^{T}} \cdot s^{M+1}, \tag{6.37}$$

where $\dot{F}^{M}(n^{m})$ is the Jacobian matrix

$$\dot{F}^{m}(n^{m}) = \begin{bmatrix} \dot{f}(n_1^m) & 0 & \dots & 0 \\ 0 & \dot{f}(n_2^m) & \dots & 0 \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ 0 & 0 & & \dot{f}(n_{S^m}^m) \end{bmatrix} \tag{6.38}$$

For the output layer M, the sensitivity s^M is 1

The sensitivity of hidden layers is computed as:

$$s^{m} = \frac{df^{m}}{dn^{m}} \cdot \left(W^{m+1^{T}} \cdot s^{m+1} \right) \;\; for\, m = 1,2,....,M-1, \tag{6.39}$$

where $\frac{df^{m}}{dn^{m}}$ is the derivative of the transfer function at layer m, and M is the number of layers. The derivative with respect to the network input is computed as:

$$\frac{\partial a^{m}}{\partial n^{1}} = W^{1^{T}} \cdot s^{1}$$

6.16 Novelty detection

Multilayer feed forward neural network can fit, in principle any real valued continuous function of any dimension to an arbitrary accuracy provided it has sufficient number of neurons (Cybenko, 1989). However, the performance of the trained neural network depends upon the data set provided during the training. The training algorithm minimizes errors between the targets and the network outputs for the training data. But this does not guarantee the neural network performance for a data that was not present in the training set, usually called as testing data.

Consider an example of approximating a function: $f(x) = x^4$. The inputs p used for the training the neural network are selected within the range [-1, 1]. Now this trained network is tested for new inputs ranging from [-2, 2]. Fig 6.8 shows a plot of neural network output trained over the range [-1, 1] and tested over the range [-2, 2].

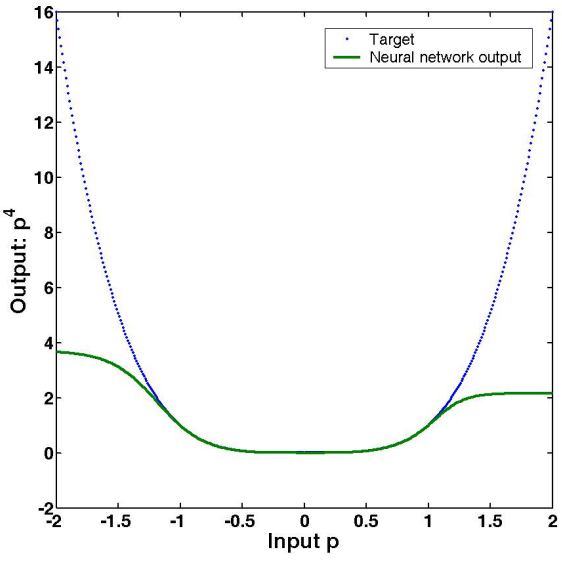


Fig 6.8 Plot of neural network output for $f(x) = x^4$

In the region of input between [-1, 1], which is same as that of training data, the network performed well. However the network performs poorly outside this region, producing very large errors between targets and outputs.

In the case of neural network fitting for a multivariate function, it becomes difficult to decide which input vector lies outside the range of the inputs in the training set and whether to count on the neural network output for a given input vector. Also, when the dimension of the input vector is large then the distinction between interpolation and extrapolation becomes more ambiguous. Therefore, differentiating the boundaries between normal and abnormal data is much more difficult, making it difficult to decide whether a specific input should be accepted or rejected. Novelty Detection is a technique that is used to identify abnormal data points. Thus, the novelty detection algorithm should be capable of identifying the inputs that fall outside the range of the training data. For example in the above example of the function of one variable, if an input is in between [-1, 1], then the corresponding output is accepted, whereas is an input is out of range [-1, 1] then the corresponding output is rejected. Some of the novelty detection techniques (Pukrittayakamee and Hagan, 2001) are:

1. Minimum distance computation
2. Neural tree algorithm (Martinez, 1998)
3. Gaussian kernel estimator (Bishop, 1994)
4. Associative multilayer perceptron (Frosini, 1996)

6.16.1 Minimum distance computation

This method is based on the 2-norm distance calculation between training vectors and the testing vector. When two vectors are close then their individual elements would also be close to each other. The 2-norm distance between the vectors is given as,

$$\left\| a_i - b_j \right\| = \left[\left(a_i - b_j \right)^T \cdot \left(a_i - b_j \right) \right]^{1/2}, \tag{6.40}$$

where a_i and b_j are the two vectors.

A vector d is constructed whose elements are distances of a testing vector to all input vectors used for training. The minimum distance separating the testing vector from all training vectors is,

$$d_m = min(d),$$

and the mean separation distance $<d>$ is the average value of the elements of d. A large value of d_m would indicate that the testing vector is different from training vectors for which the neural network was trained. Next step would be to incorporate this testing vector into a new training data set along with original training data set, and train a new neural network for this new data set. On the other hand, a smaller value of d_m suggests that there is at least one point in the training data set that is close to the testing vector, and the testing vector may or may not be included into the new training data set. This is decided based on the average distance $<d>$ as well. If $<d>$ is also small then that suggests that there are many data points in the training set close to the testing data point and this testing data point need not be included in the new training data set. In other words, the density of points close to the testing point is high. But, if d_m is small and $<d>$ is large, then it indicates that although there is at least one training data point close to the testing point, the density of points close to the testing point is small. In other words, this

point lies in an isolated region from other data points in the training set and this testing point should be incorporated into the new training set to improve the overall neural network fit.

A selection procedure based only on the magnitude of d_m may not be able to detect points that are critical to improve the fit. In regions where the gradient is large many points would be required to obtain an accurate fit. The network outputs and targets could also be used for novelty detection. The minimum distance algorithm is modified to include the outputs as well.

For an input vector p the error between the output a and target t is written as:

$$e^T = t^T - a^T = F(p^T) - a^T, \tag{6.41}$$

where F is the function to be approximated.

F could be expanded about the training input vector closet to p using Taylor's series expansion as:

$$\begin{aligned} e^T &= F(p_0) + \frac{\partial F(p)^T}{\partial p}\Big|_{p=p_0}\left(p^T - p_0\right) + \ldots \\ &= (t_0 - a_0{}^T) + \frac{\partial F(p)^T}{\partial p}\Big|_{p=p_0}\left(p^T - p_0\right) + \ldots \end{aligned} \tag{6.42}$$

where p_0 is the training vector and $\frac{\partial F(p)^T}{\partial p}\Big|_{p=p_0}$ is the first derivative of the function with respect to the input evaluated at the training vector.

Now, if p_m is the closest vector in the training set to p, with a minimum distance d_m and if higher order terms could be ignored, then

$$e^T \approx (t_m - a^T) + \frac{\partial F(p)^T}{\partial p}\Big|_{p=p_m}\left(p^T - p_m\right). \tag{6.43}$$

So, the network error not only depends on the distance between the testing vector p and the closest training vector p_m, but also on the gradient evaluated at the training vector.

Therefore, if the underlying function has a large slope, then the error could be large even though the minimum distance is small.

The underlying function is generally not known, and therefore its gradient could not be computed. Pukrittayakamee and Hagan (2001) found that an approximation of the gradient could be used if the minimum distance computation is modified to include the distance between the network output and the target. The modified vector is [*p t*], where *p* is the input vector and *t* is the network output vector.

$$d = \begin{bmatrix} \left\| p^T - p_1 \right\| + \alpha \left\| a - t_1 \right\| \\ \left\| p^T - p_2 \right\| + \alpha \left\| a - t_2 \right\| \\ \cdot \quad \cdot \\ \cdot \quad \cdot \\ \cdot \quad \cdot \\ \left\| p^T - p_N \right\| + \alpha \left\| a - t_N \right\| \end{bmatrix} \tag{6.44}$$

where α is the greater than zero, and *N* is the total number of training points.

CHAPTER 7

Ab Initio Molecular Dynamics (AIMD)

7.1 Approach

This chapter introduces an integrated approach for obtaining *ab initio* quantum mechanical potential energy surfaces and force fields for use in molecular dynamics simulations. The method involves following steps:

1. Perform MD simulations using an empirical potential which can reasonably model the interatomic interaction of the material. Store the configurations of atoms comprising of number of atoms within the cut-off radius of a particular atom and coordinates of respective atoms during the simulation. These configurations make up the initial set of configurations and serve as the starting point.
2. Perform electronic structure calculations of energies and force fields for the configurations stored in step 1 configurations using *electronic structure program* such as Gaussian 98.
3. Train a multilayer neural network to fit the scaled potential energy (obtained by shifting the zero of potential energy) and force fields for the configurations.

4. Perform MD simulation using the output from the neural network for energy and force. Observe the energy conservation and store more configurations during the simulation.
5. Compare the configurations stored while using the neural network in step 4 with those stored in step 1, and using novelty detection technique find the outlier configurations i.e. the configurations that are different from those in the training set, and form a new data base of configurations by augmenting the original data set with these new outlier configurations.
6. Perform electronic structure calculations on the new set of configurations using *electronic structure program* such as Gaussian 98.
7. Retrain the neural network for this new data base of configurations.
8. Repeat steps 4-7 until the configurations during the MD simulation are no longer different from those in the data base for which the neural network is trained. This would imply that the neural network is properly trained over the range of configuration space involved during the simulation.

7.2 Choice of the coordinate system

Cartesian coordinates provide an efficient representation of the molecular geometry for the computer, and have the advantage of including actual spatial orientation of the molecule. The position of an atom is specified using its *X, Y, and Z* coordinates. Hence to specify a molecular geometry in Cartesian coordinates, one need to specify 3N coordinates, where N is the number of atoms in the molecular system. The use of internal coordinates allows one to specify the geometry with respect to the body-fixed frame so

that translational and rotational degrees of freedom of the system can be ignored and the geometry is specified using 3N-6 internal coordinates of the molecule.

7.2.1 Internal coordinates

Internal coordinates are generally the most efficient way to describe the potential energy as a function of atomic coordinates. However since the internal coordinates are coupled, it is convenient to solve the equations of motion during MD simulation in the Cartesian coordinate system. Consequently, it becomes necessary to convert a set of Cartesian coordinates to corresponding internal coordinates. However there is no unique set of Cartesian coordinates for a given set of internal coordinates due to translational and rotational degrees of freedom involved and due to nonlinear dependence of the internal coordinates upon the Cartesian coordinates. For converting internal coordinates to Cartesian coordinates, some constraints have to be imposed such as fixing the position of the center of mass at the origin for a given configuration and aligning the coordinate axis along a specific atom (first neighbor represented as atom *2* in Gaussian 98)

Internal coordinates are specified using bond distances, bond angles and dihedral (torsional) angles. Fig 7.1 shows variables involved in internal coordinates. The dihedral angle is the angle between two planes passing through atoms *i, j, l* and *j, k, l* respectively. It is measured as the angle between the vector normal to these planes. Unlike the bond angle which varies from 0°-360°, dihedral angle varies from -180°-180°.

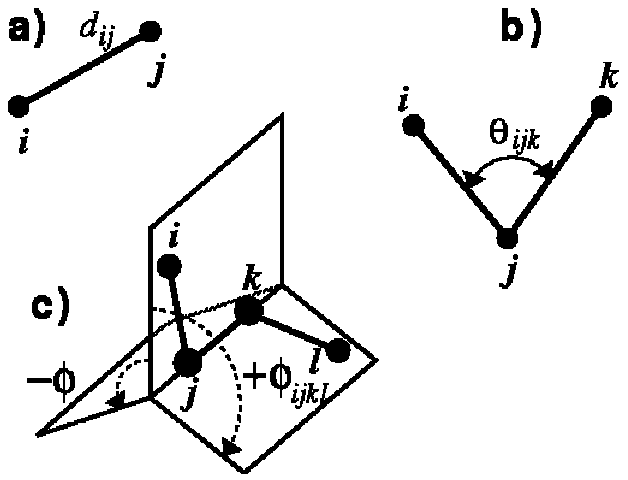


Fig 7.1 Internal coordinates a) bond distance, b) bond angle, and c) dihedral angle.

Dihedral angle is calculated as:

$$\cos \Phi_{ijkl} = (\hat{e}_{ij} \text{ X } \hat{e}_{jk}) \cdot (\hat{e}_{jk} \text{ X } \hat{e}_{kl}) / (\sin\theta_{ijk} \cdot \sin\theta_{jkl}), \tag{7.1}$$

where $\hat{e}_{ij}$ is the unit vector pointing from atom *i* to *j*.

Alternatively, if the equations of planes formed by atoms *i, j, k* and *j, k, l* are expressed as:

$$a_1 x + b_1 y + c_1 z + d \, . \tag{7.2}$$

Then the normal vector is given by $n_1 = (a_1, b_1, c_1)$, and the dihedral angle is defined as:

$$\cos\theta = n_1 \cdot n_2 = \frac{a_1 a_2 + b_1 b_2 + c_1 c_2}{\sqrt{a_1^2 + b_1^2 + c_1^2}\sqrt{a_2^2 + b_2^2 + c_2^2}} \tag{7.3}$$

The dihedral angle is either positive or negative depending on the relative rotation of the configuration with respect to an observer. Only the absolute value of the dihedral angle can be obtained using Eqn. (7.1) or Eqn.(7.3), and some sign convention has to be followed to determine the sign of the dihedral angle. Dihedral angles can be viewed using Newman projections. It may be noted that Newman projections are drawings used to represent different positions of parts of molecules with respect to each other in space. The structure is viewed along the bond between two adjacent atoms and the bonds to other

atoms in the configuration are drawn as projections in the plane of the paper. When a dihedral angle is viewed from front to rear, if the atom is clockwise from front, the dihedral angle is positive. If the rear atom is anticlockwise the dihedral angle is negative. Fig 7.2 shows the sign convention for the dihedral angle.

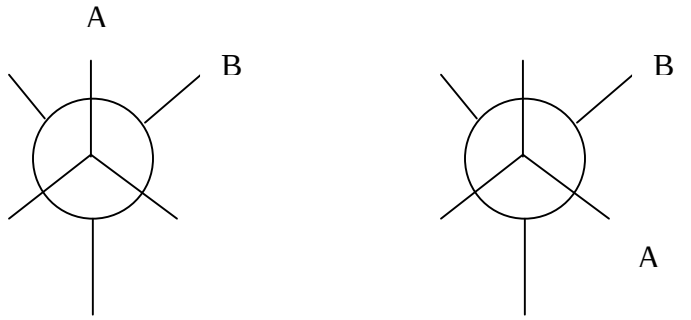


Fig 7.2 Sign convention for dihedral angle

7.3 Input vector to the neural network

A multilayer neural network is used to fit the energy or forces from the *ab initio* calculations. The input vector to the neural network should consist of variables that define the geometry of a given configuration uniquely in body fixed frame of reference.

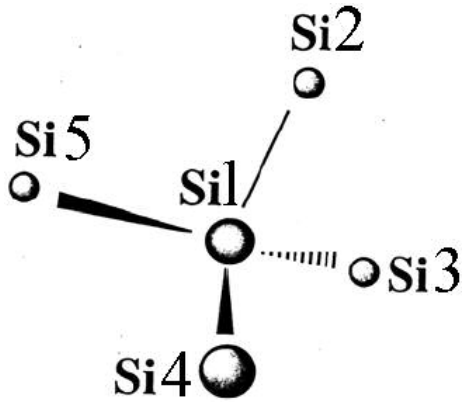


Fig 7.3 Configuration of 5 silicon atoms

Fig 7.3 shows a typical configuration of 5 silicon atoms where Si1 is the central atom and atoms Si2, Si3, Si4 and Si5 are bonded to the Si1 atom. The dotted line for bond Si1-Si3 shows that the atom Si3 is projecting inside the plane of the paper, while solid lines for bonds Si1-Si4 and Si1-Si5 show that the atoms Si4 and Si5 are projecting out of the plane of the paper, while atom Si1 and Si2 are in the plane of the paper.

Now the input vector to the neural network for the configuration shown in Fig 7.3 could be constructed in two ways. One approach would be to form the input vector consisting of all the bond distances. For example the input vector could be formed as:

$$in_1 = [r_{12}; r_{13}; r_{14}; r_{15}; r_{23}; r_{24}; r_{25}; r_{34}; r_{35}; r_{45}],$$

where r_{12}, r_{13}, r_{14}, and r_{15} are bond distances between atoms Si1 and Si2, Si3, Si4 and Si5 respectively. Such a vector would uniquely specify the geometry of a given configuration. In this case, the number of variables involved would be $n\ (n-1)/2$, where n is the size of the cluster (for a cluster of 5 silicon atoms, $n=5$). Other approach would be to form the input vector consisting of internal coordinates. This formulation would require $3n - 6$ variables as opposed to $n\ (n-1)/2$ variables as in the case of input vector consisting of all bond distances. The configuration in Fig 7.3 can be written in terms of internal coordinates using 4- bond distances, 3- bond angles and 2- dihedral angles as:

$$in_2 = \left[r_{12}, r_{13}, r_{14}, r_{15}, \theta_{312}, \theta_{412}, \theta_{512}, \Phi_{4123}, \Phi_{5124}\right],$$

where r_{12}, r_{13}, r_{14}, r_{15} are bond distances between atoms *Si1* and *Si2*, *Si3*, *Si4* and *Si5*, respectively, and θ_{312}, θ_{412}, θ_{512} are bond angles that atoms *Si3*, *Si4* and *Si5* make with bond 1-2, and Φ_{4123}, Φ_{5124} are the dihedral angles. While measuring the bond angles and dihedral angles bond 1-2 (Si1-Si2) is used as a reference line. So that the input vector

so constructed is consistent with the convention followed in Gaussian 98, which places the Z-axis along bond 1-2 while using internal coordinates.

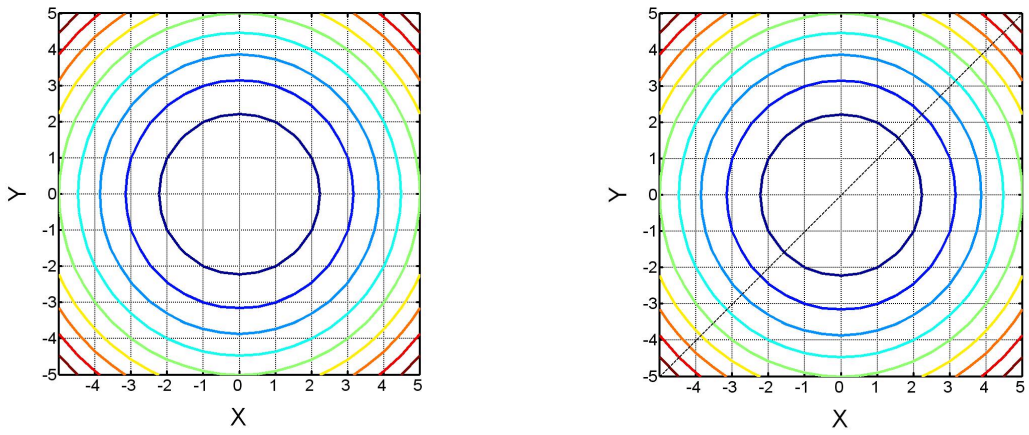
It should be noted that the length of the input vector formed using all bond distances varies as the square of *n* i.e. the cluster size, whereas the length of the input vector formed using internal coordinates varies linearly (three times) with the cluster size. As a result, for the cluster size of five or greater than five, the input vector formed using the internal coordinates would involve less number of variables compared to the input vector formed consisting of all bond distances. The use of internal coordinates reduces the dimension (length) of the input vector used for training the neural network. Consequently, the input space is less sparse. This could result in faster convergence during training with less number of training configurations. This also reduces the overall complexity involved in mapping the input variables to the output variables.

7.4 Sorting and ordering the input vector for neural network training

Consider a function of two variables: $f(x,y) = x^2+y^2$, which is a paraboloid. Fig 7.4 shows the contour plot of the function. Since the function is parabolic so the contours are circles, and each circle represents a constant function value curve, i.e. the points on a contour represent a set of [*x*,*y*] values which result in the same value for *f*. It should be noted that $f\ (2,4) = f\ (4,2)$, since points [2,4] and [4,2] are on the same contour corresponding to $f = 20$.

Now if a neural network is to be trained for this function of two variables (*x*,*y*) then the input vector corresponding to f = 20 could be specified as either $\begin{bmatrix}2\\4\end{bmatrix}$ or $\begin{bmatrix}4\\2\end{bmatrix}$. The difference between these two input vectors is the different ordering of the variables *x*

and *y*. It is possible to reduce the configuration space required to be trained if the symmetry involved in specifying the input vector is taken into consideration. The input could be ordered so that the first element is *x* and the second element is *y* (or vice versa). So instead of training the network with two inputs differing only in their ordering and having the same output, the network could be trained for only one ordered input vector. Fig 7.4 (b) shows the contour plot of the function, the diagonal line separates the two input vectors having the same function value but differing only in their order of representation.



(a) (b)

Fig 7.4 Contour plot of a function $f(x,y) = x^2+y^2$

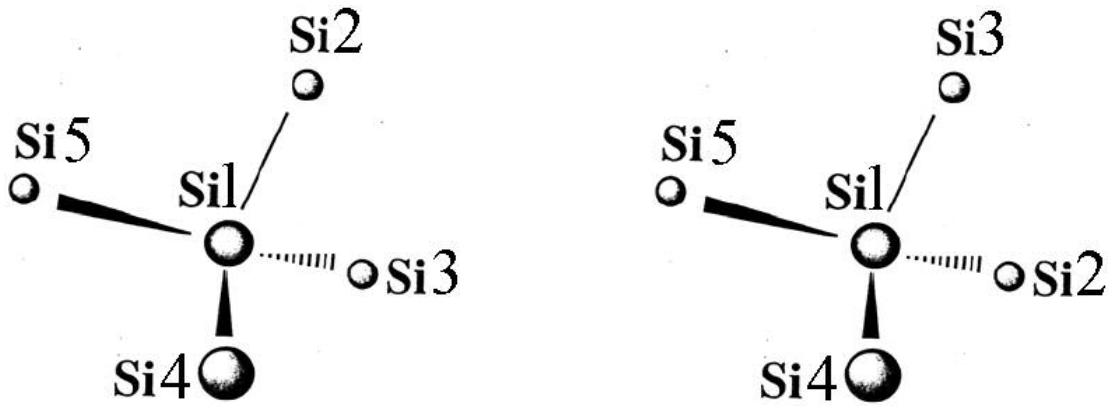
As shown in Fig 7.4 (b) either the upper or the lower region about the diagonal need to be considered for training. Thus for a two variable function, by ordering the input vector to the network reduces the configuration space required for training by a factor of half. Such an approach proves to be more efficient in a situation where the input vector

dimension is large and the number of data points is limited. This results in sparse data points in high dimension making the network training difficult.

7.4.1 Procedure for sorting and ordering the input vector for AIMD

It is important to take the symmetry of the configuration into account. For the electronic structure program, such as Gaussian 98 the energy and force fields for a given Si_5 cluster are unaltered if two or three silicon atoms are exchanged. However, the neural network training algorithm does not take into account the symmetry involved in the input vector.

Therefore, the neural network must be trained to recognize this symmetry or a different procedure for constructing the input vector should be adopted that makes this unnecessary in order for the neural network to recognize different ordering of atoms within the same configuration. Consider a configuration as shown in Fig 7.5. Both (a) and (b) show the same configuration but with different ordering of atoms Si2 and Si3.



(a) (b)

Fig 7.5 Effect of ordering of atoms in a configuration

The configurations shown in Fig 7.5 (a) and (b) both have the same energy and forces along the bonds and either of the two could be used for *ab initio* calculations. But the two configurations will have different input vectors for neural network training because bond used as the reference line for measuring the angles is different in the two cases, 1-2 in case (a) and 1-3 in case (b). The input vectors for these two configurations could be written as:

$$\text{in}_{(a)} = \left[r_{12}, r_{13}, r_{14}, r_{15}, \theta_{312}, \theta_{412}, \theta_{512}, \Phi_{4123}, \Phi_{5124}\right],$$

$$\text{in}_{(b)} = \left[r_{13}, r_{12}, r_{14}, r_{15}, \theta_{213}, \theta_{413}, \theta_{513}, \Phi_{4132}, \Phi_{5134}\right].$$

For any non-equilibrium geometry, different angles are involved in the input vector. So, for the neural network the configurations in Fig 7.5 (a) and (b) are two different inputs. For proper generalization of the fit, the neural network has to be trained for both the input vectors having the same energy as the output. For a given configuration there are (n-1)! symmetries (permutations) possible the with same energy and force fields, where n is the size of the cluster. As a result the number of configurations required for training the neural network increases exponentially as the number of atoms within a cluster increases and it becomes unfeasible to consider all the symmetries. For example: if there are N different configurations for which *ab initio* calculations are performed then for the neural network training there would be N(n-1)! data points considering all symmetries. For a cluster of Si_5 atoms (n-1)! is 24, hence with 10,000 different configurations from Gaussian 98 there would be 240,000 data points for neural network training. Considering all possible permutations of the same configuration proves to be an inefficient approach for the neural network training, since it not only increases the

number of configurations required for training by a factor of (N-1)! but also increases the configuration space and complexity of the function to be trained.

To circumvent this problem it becomes necessary to devise a strategy for training the neural network without considering the permutations. One way to achieve this is to order the input vector which obviates the need to consider all permutations without affecting the generalization ability of the neural network. Consider a configuration as shown in Fig 7.3. First calculate all bond distances with the central atom Si1, i.e. r1-2, r1-3, r1-4 and r1-5, sort these bond distances and order the atoms Si2, Si3, Si4 and Si5 and then calculate the angles with bond 1-2 as the reference line (2 corresponds to atom 2 after sorting and ordering). The ordering of the input vector circumvents the need to consider all permutations and also reduces the configuration space of the input variables required for training and hence the overall complexity of the function involved. As a result time required for training the neural network is reduced. Also a simpler neural network architecture (having less number of neurons) could be implemented and the accuracy of the fit could be improved for a given number of neurons. The same procedure should be followed during the MD/MC simulation to construct the input vector, in order to calculate the energy and forces using the trained network. This procedure results in a slight increase in the computational time during the simulation. But the advantages offered by sorting and ordering the input vector offset this slight increase in the computational time.

7.5 Calculation of forces during the MD simulation

Molecular dynamics simulation involves numerical integration of the classical Newton's equations of motion for a system of interacting atoms over a period of time. According to Newton's second law:

$$F_i = m_i\, a_i \tag{7.4}$$

where F_i is the force on atom *i*, and m_i and a_i are the mass and acceleration of atom *i*. The force is the negative gradient of the potential energy with respect to the position of an atom.

$$\vec{F}_i = -\nabla_i V(\vec{r}_1, \vec{r}_2, ...\vec{r}_N), \tag{7.5}$$

where V is the potential energy function. $\vec{r}_i$ is the position vector of atom i, x_i, y_i and z_i, are the Cartesian coordinates of atom i.

$$\vec{r}_i = x_i\,\hat{i} + y_i\,\hat{j} + z_i\,\hat{k}, \tag{7.6}$$

$$\nabla_i = \frac{\partial}{\partial x_i}\hat{i} + \frac{\partial}{\partial y_i}\hat{j} + \frac{\partial}{\partial z_i}\hat{k}. \tag{7.7}$$

The force on an atom could be computed in two ways. One approach would be to first train a neural network for the forces and use this network in the simulation. Now for a cluster of *n* atoms there would be at least *3n-6* variables required to define the geometry uniquely. If a neural network is to be trained for the forces then such a network would have *3n-6* input variables (bond distances and angles) and also *3n-6* output variables (forces along the bond distances and angles). It would be a very demanding task to train a single network to perform this, since the functional dependence of angular forces on the input variables may be very different from the functional dependence of forces along the bonds. It would be like trying to fit different functional relationships for a given set of input variables. Although such a network could be trained with a large number of neurons and consequently more training time, a better approach would be to train the network for the potential energy of the molecular cluster and then during the simulation differentiate

this network with respect to the input variables to get the forces. For a specified accuracy such a network may require less number of neurons than a network which tries to directly fit all the derivatives. This reduces the training time but on the other hand during the simulation the output from the neural network has to be differentiated to calculate the forces which introduce extra calculations during the simulation. The approach involving the neural network differentiation would be feasible only if the time required for the differentiation of a possibly a smaller network is lesser than the time required to compute directly the derivatives from a larger network with comparable accuracy. Since the neural network involves analytical differentiable transfer functions, such as sigmoid functions for mapping, it could be differentiated analytically without introducing any numerical errors. For the multilayer network, the network output is given as:

$$a^{m+1} = f^{m+1}(W^{m+1}a^{m} + b^{m+1}) \;\; for\, m = 0,1,...,M-1, \tag{7.8}$$

where the superscript represents the layer number. M is the number of layers of the network, and input to the first layer is $a^0 = p$

The sensitivity is defined as:

$$s^{M} = \frac{da^{M}}{dn^{M}} = \dot{F}^{M}(n^{m}) \cdot W^{M+1^{T}} \cdot s^{M+1}, \tag{7.9}$$

where $\dot{F}^{M}(n^{m})$ is the Jacobian matrix

$$\dot{F}^{m}(n^{m}) = \begin{bmatrix} \dot{f}(n_1^m) & 0 & \dots & 0 \\ 0 & \dot{f}(n_2^m) & \dots & 0 \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ 0 & 0 & & \dot{f}(n_{S^m}^m) \end{bmatrix} \tag{7.10}$$

For the output layer M, the sensitivity s^M is 1

The sensitivity of hidden layers is computed as:

$$s^m = \frac{df^m}{dn^m} \cdot \left(W^{m+1^T} \cdot s^{m+1} \right) \quad for\ m = 1,2,....,M-1, \tag{7.11}$$

where $\frac{df^m}{dn^m}$ is the derivative of the transfer function at layer *m*, and *M* is the number of layers.

The derivative with respect to the network input is computed as:

$$\frac{\partial a^m}{\partial n^1} = W^{1^T} \cdot s^1. \tag{7.12}$$

7.6 Sampling technique

For a typical configuration of 5 silicon atoms as shown in Fig 7.3, the input vector using internal coordinates is specified as:

$$\left[r_{12}; r_{13}; r_{14}; r_{15}; \theta_{312}; \theta_{412}; \theta_{512}; \Phi_{4123}; \Phi_{5124} \right].$$

The neural network is trained for energy computed from Gaussian 98 as a function of nine variables in internal coordinates. Although it is possible to compute energies and forces at MP4 (SDQ) and DFT level for thousands of configurations, this number of configurations is still small compared to the totality of the configuration space available for the system. If reasonable ranges are taken for each coordinate and each range divided into ten equal segments, then the total number of possible configurations would be 10^9. This number of configurations is impractical considering the present computational capabilities. However because of bonding restrictions present in the system not all of the configuration space is of interest in the dynamical studies. For example, in the simulation

of bulk silicon material, configurations such as Si-Si-Si-Si-Si i.e. five silicon atoms in a linear chain are of no interest. In any system, the number of important configurations forms a small subset of the total configuration space. Therefore, the key is to sample only the important region of the total configuration space.

Since an empirical potential is used during the first step to store the configurations, this set of configurations need not necessarily cover the entire configuration space of the *ab initio* potential energy. This empirical potential need not be highly accurate but should provide a reasonable representation of the interatomic interaction of the material. The configurations that are generated in an ensemble of trajectories using an empirical potential comprise the subset of configuration space that has to be sampled. The procedure that has been employed attempts to sample the entire configuration space in an iterative fashion. A set of trajectories using this empirical potential is computed. Next the energy and forces for these configurations are computed using Gaussian 98. This data makes up the first approximation for the *ab initio* potential energy and force field. Then a neural network is trained to the *ab initio* database with the input vector consisting of internal coordinates and the output being the energy or force components. Now a second set of MD trajectories is computed using this neural network force field rather than the original empirical potential. During these calculations atomic configurations are stored and their energies and forces are computed using Gaussian 98. After several iterations neural network is expected to converge to *ab initio* potential energy and force field.

Given a sufficient number of neurons, a neural network can be trained to approximate any arbitrary function (Cybenko, 1989). However, the performance of the

trained neural network will be dependent upon the data set that was provided during the training period. A training algorithm minimizes the errors between the targets and the neural network outputs for the training data set. But this does not guarantee the neural network performance for a data that was not present in the training set, usually called as testing data. In the case of neural network fitting, for a multidimensional variable function, it becomes difficult to decide which input vector lies outside the range of the training set. Also when the dimension of the input vector is large then the distinction between interpolation and extrapolation becomes more ambiguous. Therefore, differentiating the boundaries between normal and abnormal (outlier) data is much more complicated, making it difficult to decide whether a specific input should be accepted or rejected.

7.7 Novelty detection

Novelty Detection is a technique that is used to identify abnormal data points. Thus, the novelty detection algorithm should be capable of identifying the inputs that fall outside the range of the training data. Some of the novelty detection techniques (Pukrittayakamee and Hagan, 2001) are:

5. Minimum distance computation
6. Neural tree algorithm (Martinez, 1998)
7. Gaussian kernel estimator (Bishop, 1994)
8. Associative multilayer perceptron (Frosini, 1996)

In the context of *AIMD*, the novelty sampling algorithms based on the current density of configuration points in the data base guide the selection of configurations to be included

to form a new data base. Regions with a low density of configurations are preferentially included in the sample.

Let q_i be an input vector corresponding to the i^{th} configuration, in the current data base. By computing trajectories using the neural network output for the force field a new configuration q_n is generated. Qualitatively this configuration should be included in the data base if q_n lies in a region of space where the density of points in the data base is low. Minimum distance computation is used for the novelty detection. It is based on the 2-norm difference between vector q_n and vector q_i:

$$\|q_i - q_n\| = \left[(q_i - q_n)^T \cdot (q_i - q_n)\right]^{1/2}. \tag{7.13}$$

A vector d is now defined whose elements are all the distances computed using Eqn.(7.13) for each of the N configuration points in the data base. The minimum distance separating point q_n from other points in the data base is the minimum element of d denoted as d_m=min(d). The mean separation distance $<d>$ is the average value of the elements of d. The configuration point q_n will be added to the data base with a high probability if d_m or $<d>$ is large, but with low probability if both d_m and $<d>$ are small.

A selection algorithm based solely upon the magnitude of d_m has a drawback that it will not detect the points that are needed in the database, in the region where the configuration density is not low. This is the case when the function is sharply changing. Hence, in the regions where the gradient of the function is large, many points will be required to obtain an accurate neural network fit than in the case with smaller gradient, even though the configuration density is not low. But the underlying function is unknown. Therefore it is impossible to calculate the gradient at an arbitrary configuration point. Pukrittayakamee and Hagan (2001) have found that if the minimum distance

between a set of modified configuration vectors is used in place of d_m, then the effect of the function gradient can be incorporated into the selection algorithm. A modified vector is created as:

$$\Gamma_i = [q_i \; O_i], \tag{7.14}$$

where O_i is the output from the neural network of the configuration point q_i. When the minimum distance Γ_m *between* Γ_i and Γ_n is used in place of d_m, the effect of function gradient is incorporated. This is because, if the gradient between points q_i and q_n is large, then the difference between corresponding neural network outputs O_i and O_n will be large, making the magnitude of Γ_m large, and hence the point q_n will be identified as a novelty point.

CHAPTER 8

Results and Disscussion

In the present study the approach was used to develop potential energy hypersurface for a cluster of five silicon atoms. *Ab initio* calculations were performed using Gaussian-98 electronic structure program using density functional theory. A feed-forward multilayer neural network was trained using Matlab. The accuracy of the trained neural network was iteratively improved using modified novelty detection technique.

The equilibrium configuration of silicon is tetrahedral in which every silicon atom is bonded to four other silicon atoms in a tetrahedral arrangement as shown in Fig 8.1.

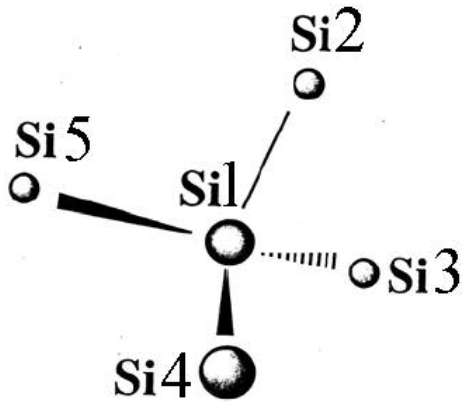


Fig 8.1 Silicon cluster

To model this system a cluster of at least five silicon atoms should be considered including the central atom and four other atoms bonded to it. MD simulations of machining of silicon workpiece were performed. Tersoff potential (Tersoff, 1989) was

employed during the initial step of the iterative process. Very high energies are present during the machining process which results in atomic configurations far off from the equilibrium tetrahedral structure especially in the chip and near the tool because of the subsurface deformation. This increases the size of subset configuration space that is sampled.

MD simulations of machining were performed with +15° rake angle, 10° clearance angle, 1Å depth of cut at a cutting speed of 491.2 ms^{-1}. The total number of atoms in the workpiece was 1,675. Throughout the MD simulations, the configurations of the five-atom silicon clusters that are present predominantly in the chip in front of the tool in the shear zone and within a few unit cell distances in the workpiece beneath the tool are stored

8.1 *Ab initio* calculations

The energies and force fields at the central atom are computed for 18,000 configurations using Gaussian 98 electronic structure program for density functional theory (DFT) calculations with 6-31G** basis set and B3LYP procedure for incorporating the correlation. The input was specified in internal coordinates i.e. *Z-matrix* format. It is important to take symmetry into account, since merely switching the ordering of atoms does not change the resulting energy and force fields from *ab initio* calculations. The neural network should also be trained to recognize different possible ordering of the same configuration involved. One option to achieve this is to train the neural network considering all possible permutations of a configuration. For a cluster of *n* atoms there are (*n-1*)! possible permutations. So with 18,000 five atoms' silicon clusters this would require. 18,000 · (5-1)! i.e. 432,000, data points to be considered for the neural

network training, which becomes unfeasible for neural network training. An alternative is to order the input vector which obviates the need to consider all permutations without affecting the generalization ability of the neural network. The *Z-matrix* input describing the Si_5 clusters comprise of four Si-Si bond distances, three Si-Si-Si- angle, and two dihedral angles. Before each calculation, the four bond distances were listed in increasing order of their deviation from the Si-Si equilibrium bond distance. This listing determines the *Z-matrix* input for the bond angles and dihedral angles. Bond Si1-Si2 was considered as reference for measuring the angles. This procedure obviates the need to train the neural network to recognize symmetry and simplifies the training. Fig 8.1 shows a typical configuration of five silicon atoms. The variables involved in *Z-matrix* input are r_{12}, r_{13}, $r_{14,}$ $r_{15,}$ θ_{312}, θ_{412}, $\theta_{512,}$ φ_{4123} and φ_{5124}, where r_{12}, r_{13}, $r_{14,}$ r_{15} are the bond distances of silicon atom #1 with atoms 2, 3, 4 and 5 respectively, θ_{312}, θ_{412}, θ_{512} are angles of that bond 1-3, 1-4 and 1-5 make with the bond 1-2 and φ_{4123} and φ_{5124} are the two dihedral angles. When the trained neural network is utilized for computing energy and force fields, the same ordering of the input vector was used.

The database of configurations obtained from the machining simulations was augmented with additional 10,000 five-atom silicon configurations near the equilibrium. These configurations were obtained corresponding to a temperature of 300 K in the silicon workpiece and following the vibrational motions of the lattice using molecular dynamics with Tersoff potential. The configurations were stored at equally spaced time intervals during the calculations. The energies and force fields for these configurations were also computed using Gaussian 98 for density functional theory (DFT) calculations with 6-31G** basis set and B3LYP procedure for incorporating the correlation.

8.2 Neural network training

Next step is to fit the neural network to the *ab initio* energy. The forces can be calculated during the MD simulation by differentiating the neural network output i.e. the energy with respect to the inputs i.e. the internal coordinates. The zero of the energy is shifted so that the energy of five infinitely separated silicon atoms corresponds to zero energy. Consequently, the energy of a cluster of five silicon atoms would be negative, which is the convention followed for empirical potential as well. The inputs i.e. the internal coordinates and outputs i.e. the energy are scaled so that the range for each variable in the entire database is [-1, +1], using Eqn.(8.1)

$$p_n = 2 \cdot \frac{(p - \min p)}{(\max p - \min p)} - 1, \tag{8.1}$$

where p is the variable to be scaled, *minp* and *maxp* are the minimum and maximum values of each variable in the input and output vectors for the entire database consisting all configurations. p_n is the scaled value corresponding to p. The scaled database is divided into three sets: training, validation and testing set. Approximately 10% of the data points were used for the validation set, 10% for the testing set and the rest for the training set. A multilayer feed forward neural network was used to fit the data. The number of neurons in the input and output layers is determined by the number of input and output variables, respectively.

The neural network specifications are listed in Table 8.1.

Table 8.1 Neural network specification for silicon clusters

Number of layers:	2, (one hidden layer and one output layer)
Number of neurons:	9 for the input layer, corresponding to the number of input variables. 45 for the first hidden layer. 1 for the output layer, corresponding to the number of output variables.
Transfer function	Hyperbolic tangent function in the hidden layer. Linear function in the output layer.
Training algorithm	Levenberg-Marquardt along with Bayesian regularization.

Designing a network involves deciding upon the number of neurons in the hidden layer. The network should have enough parameters to learn the underlining function. On the other hand, if there are too many parameters available for fitting a relatively simple function or if the number of data points is far less than the number of parameters, then the network could overfit the data i.e. it would try to fit every data point very closely while not generalizing well the configuration space. Generally, the underlined function is unknown which makes it difficult to decide the number of neurons. To avoid overfitting it is essential to employ an early stopping procedure (Sarle, 1995). In this method, the root mean square deviation of the neural network output from the data in the validation set is computed after each iteration during training. Initially, the root mean square error for both the training and validation sets decrease because the accuracy of the fit is improving. At some point during training overfitting the data can start. This would improve the apparent accuracy for the training set, but increase the accuracy for the validation and testing sets since these points are not included in the training set. For optimum performance from the specific neural network architecture, the training should be continued until the error for the validation set is a minimum value. Along with early

stopping, regularization is also used which penalizes the network model with high curvature, since high curvature normally results while overfitting the data. Bayesian regularization provides an approach to determine the optimal regularization parameter in an automated fashion. In this framework the weights and biases are assumed to be random variables with a specified distribution.

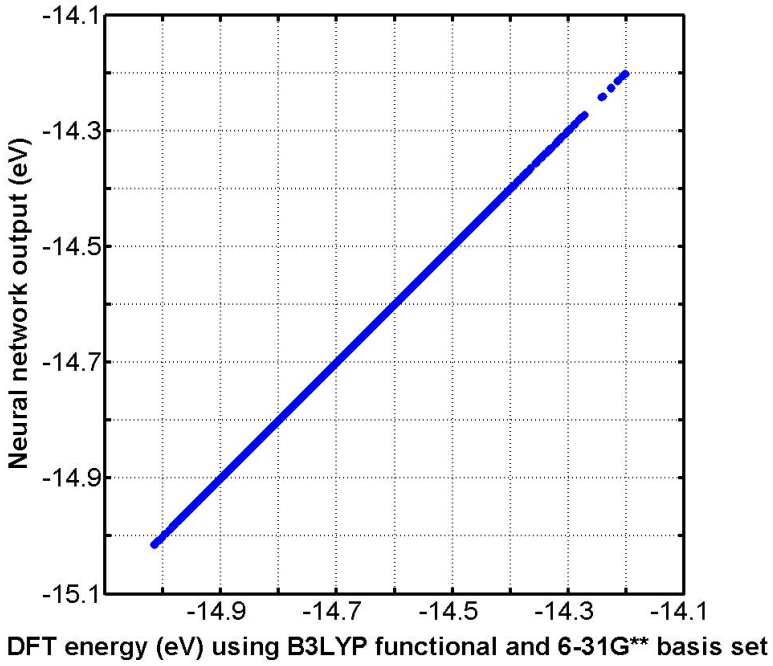


Fig 8.2 Comparison of neural network output with *ab initio* energy for the training set

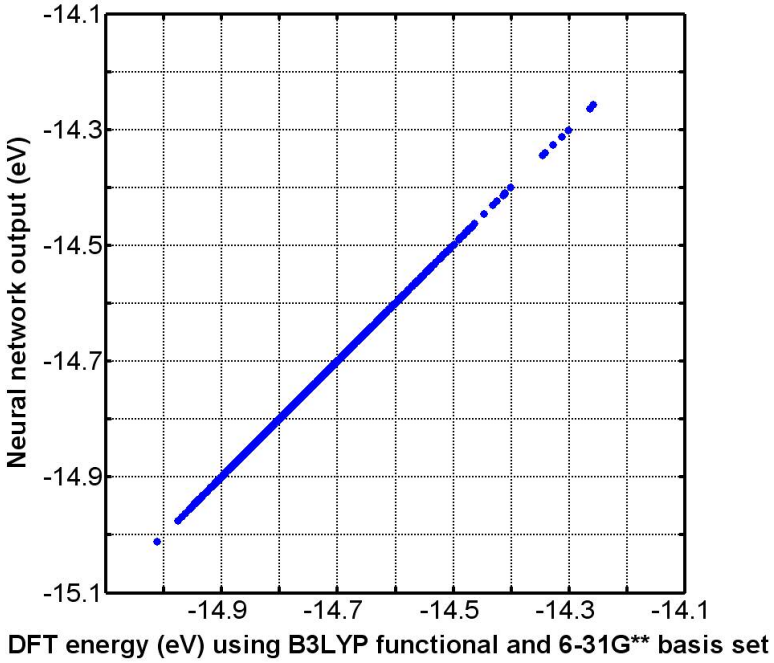


Fig 8.3 Comparison of neural network output with *ab initio* energy for the validation set

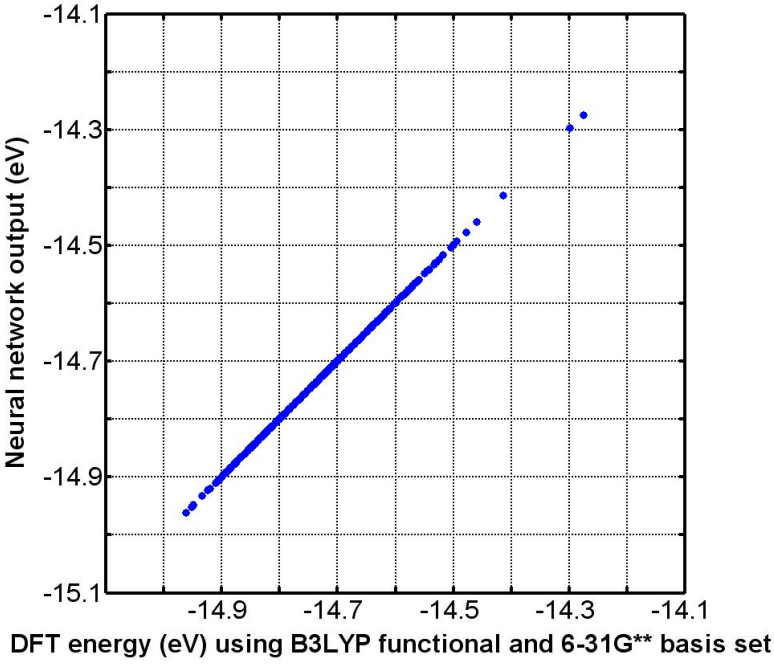


Fig 8.4 Comparison of neural network output with *ab initio* energy for the testing set

Fig 8.2 to Fig 8.4 show the plots of neural network output for the training, validation and testing sets, respectively. If the neural network fit were perfect then all points would fall along the 45° line. The computed root mean square deviation of the neural network fit from the *ab initio* energy values was 0.36 KJ mol^{-1}, which is comparable to the chemical accuracy involved in the experimental measurements.

8.3 Iterating for the neural network convergence

The minimum distance of a configuration in the initial set (consisting of configurations stored form Tersoff potential) from all other configurations in the same set is calculated. Fig 8.5 shows the distribution of minimum distances $N(\Gamma_m)$ for the initial set of Si_5 configurations from Tersoff potential. The distribution is close to Gaussian with the most likely normalized minimum distance being about 0.095.

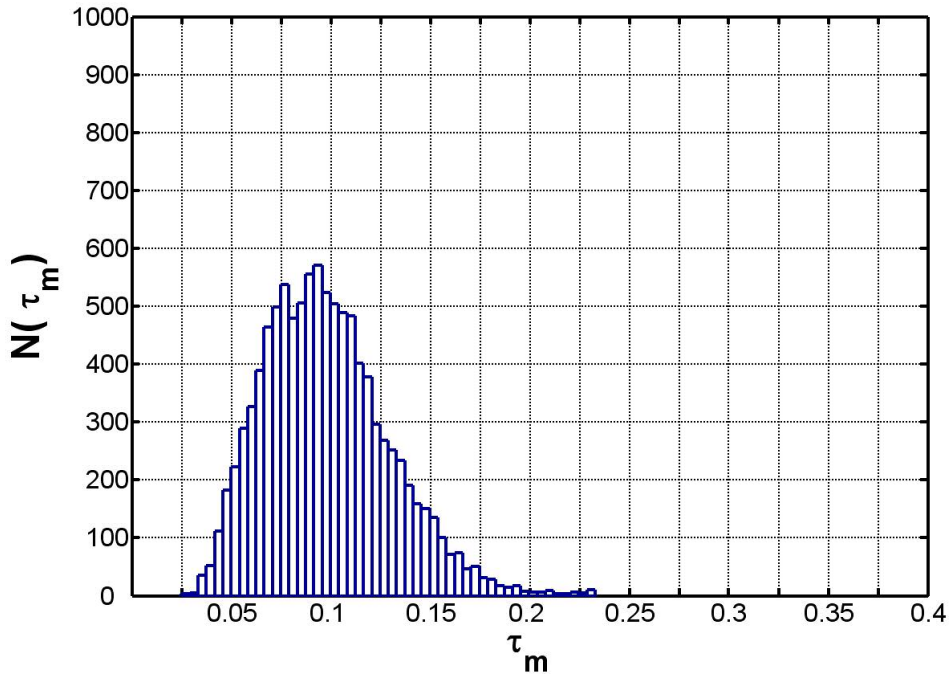


Fig 8.5 Distribution of minimum distances $N(\Gamma_m)$ for the initial set of Si_5 configurations using Tersoff potential

Next step is to use the trained neural network in MD simulation for energy and force calculations. While performing the MD simulations using the neural network, the novelty detection is also employed to obtain additional configurations to iteratively improve the potential energy hypersurface. Now the minimum distances of configurations in the new set (consisting of configurations stored during the MD simulation using neural network) are calculated from the configurations in the initial set (using Tersoff potential) and the distribution is plotted in the histogram in Fig 8.6. The distribution has a peak close to 0.25

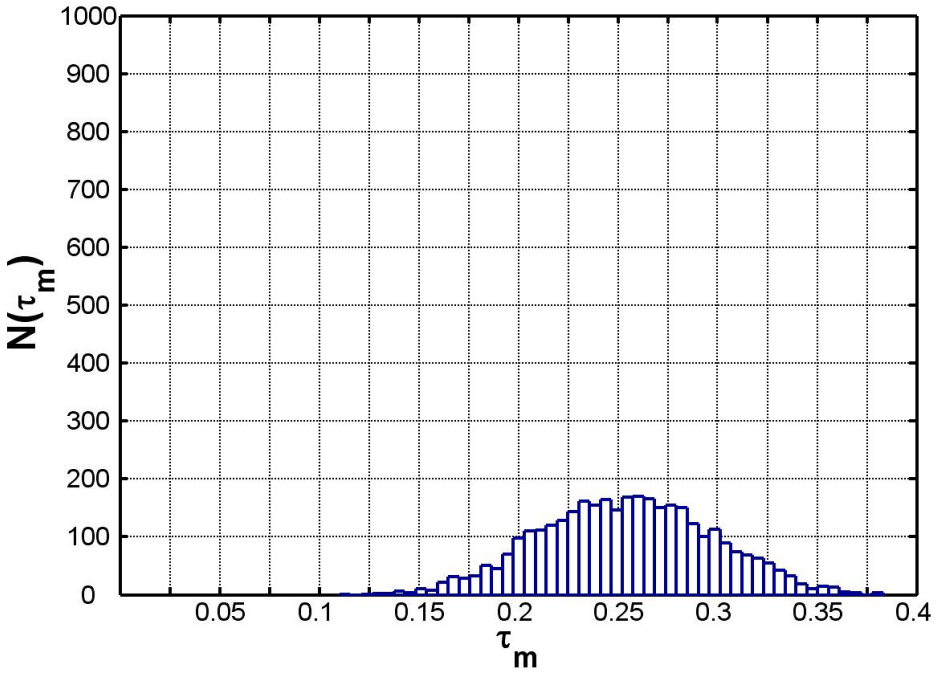


Fig 8.6 Distribution of minimum distances $N(\Gamma_m)$ of the configurations in first iteration from initial set configurations.

Fig 8.7 shows a comparison of the distribution of minimum distances $N(\Gamma_m)$ of the configurations from first iteration with the distribution of configurations from Tersoff potential. The dashed lined distribution corresponds to minimum distance of the initial

configurations and the solid lined distribution corresponds to minimum distance of configurations from the first iteration. There is very little overlap between the two distributions indicating that the configurations occurring during the MD simulation while using the neural network trained *ab initio* potential energy hypersurface are significantly different from those in Tersoff potential. This indicates that the new configurations lie in a region of configuration space not adequately sampled by the initial data set using Tersoff potential.

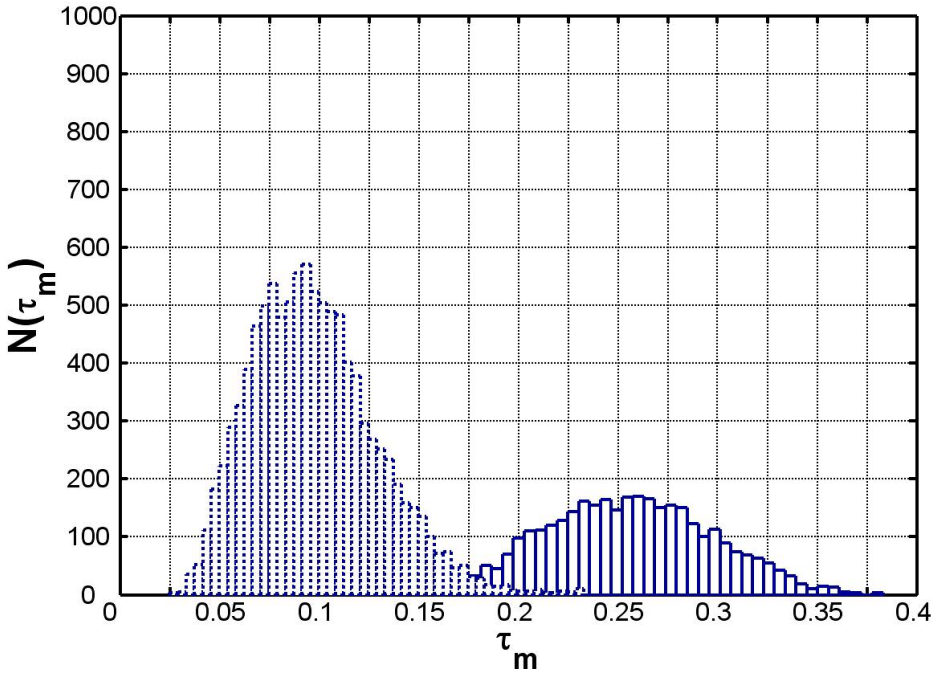


Fig 8.7 Comparison of distribution of minimum distances $N(\Gamma_m)$ of the configurations in first iteration from initial configurations with the distribution of configurations from Tersoff potential.

A new set is formed by concatenating the initial configurations and configurations during the first iteration and the minimum distance distribution is recomputed for the concatenated set. Fig (8.8) shows the distribution of recomputed minimum distances for the concatenated set. The spread of the histogram in Fig (8.8) is smaller that in Fig 8.7

because now the neural network is trained for outlier configurations as well along with initial set of configurations and as a result corresponding output is close to the target values. Also the configurations in the two sets could be overlapping, which reduces the spread of minimum distances for the concatenated data set.

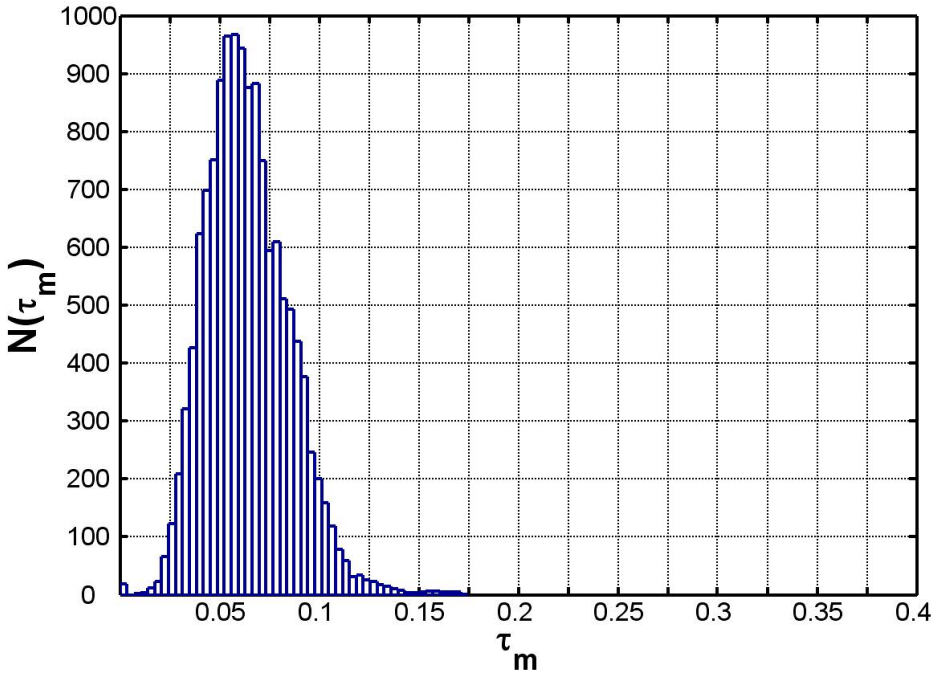


Fig (8.8) Distribution of recomputed minimum distances of concatenated set of configurations in the initial and first iteration.

Now for the selection process based on minimum distance criterion, normalized probability value should be used. For this purpose, Parzen's distribution for multivariate kernel is used (Specht, 1990). In the case of Gaussian kernel, the multivariate estimate is expressed as:

$$F_A(X) = \frac{1}{(2\pi)^{p/2}\sigma^p}\frac{1}{m}\sum_{i=1}^{m}\exp\left[-\frac{(X - X_{Ai})'(X - X_{Ai})}{2\sigma^2}\right], \tag{8.2}$$

m= Total number of configurations

X_{Ai}= minimum distance for the ith configuration.

σ= smoothing parameter

p= dimensionality of the data, since in this case there is only one number corresponding to the minimum distance so p=1.

A small value of σ causes the estimated density function to have distinct modes corresponding to the locations in the samples. A larger value of σ produces a greater degree of interpolation between points. Fig 8.9 shows a plot of Parzen's distribution for the initial set of configurations from the Tersoff potential using σ = 0.01. Fig 8.10 shows a plot of Parzen's distribution for configurations stored while running MD simulation using neural network.

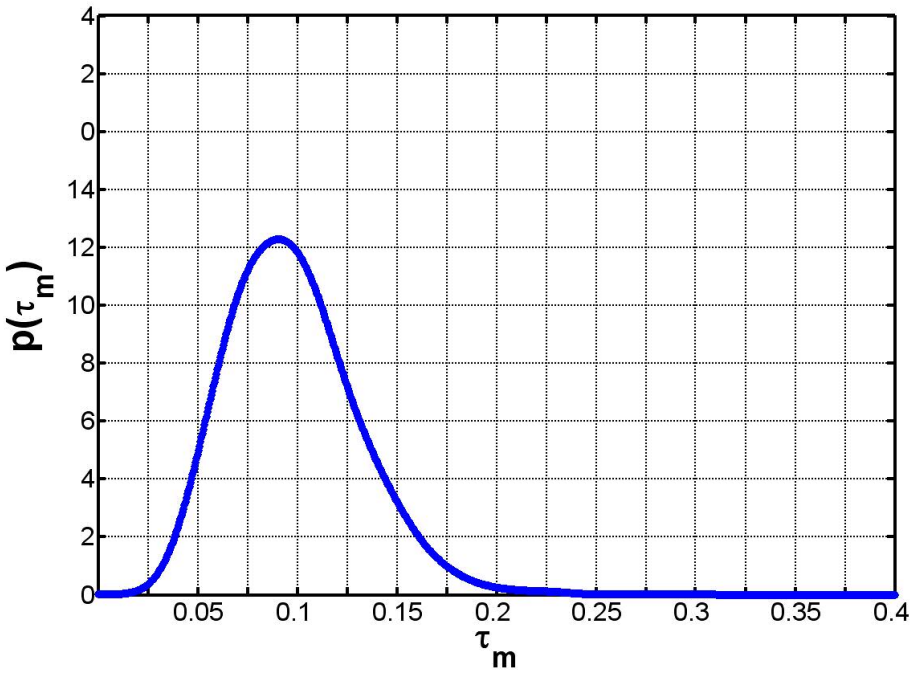


Fig 8.9 Parzen's distribution of minimum distances for the initial set of configurations from the Tersoff potential

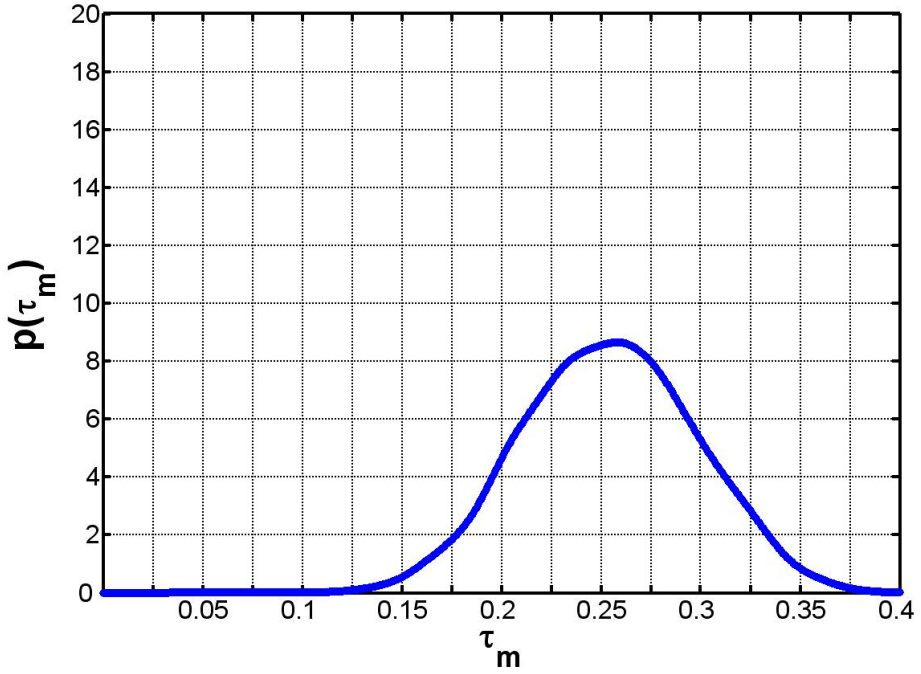


Fig 8.10 Parzen's distribution of minimum distances for the configurations stored while using neural network trained for *ab initio* energy during MD simulation.

As new configuration points are generated during the trajectories computed using the neural network force field, they are added to the data base if:

$$P(\Gamma_m) \leq T_1\, P(\Gamma_m)_{max},$$

$$\text{or, if } \frac{P(\Gamma_m)}{P(\Gamma_m)_{max}} \leq \xi^2\left(1-T_1\right) \text{ and } \Gamma_m \geq \Gamma. \tag{8.3}$$

T_1 is an empirically determined acceptance threshold n the range ($0 \leq T_1 \leq 1$) and ξ is a random number whose distribution is uniform on the interval [0,1].

Now, by examining Fig 8.9 it can be seen that for a given probability there are more than one Γ_m values, as a result when the minimum distance between the configurations Γ_m is very small and close to zero (which is the case if the two configurations are almost identical), then the corresponding probability is small and the configuration will be

accepted, which is not required since there is already at least one configuration in the set which is close to it. To avoid this in the second condition in equation (8.3), $\Gamma_m \geq \Gamma$ is used so that a configuration with low probability is accepted only if its minimum distance Γ_m is greater than some threshold value Γ.

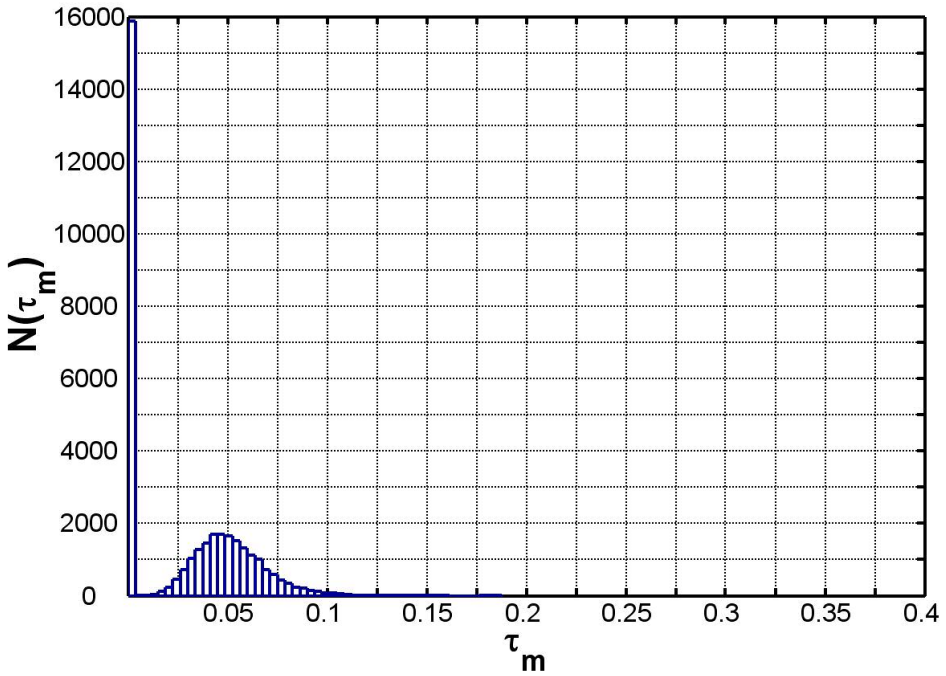


Fig 8.11 Distribution of minimum distances of the configurations during second iteration from configurations during the first iteration

Fig 8.11 shows the distribution of minimum distances of the configurations during second iteration from configurations during the first iteration. A sharp peak close to zero indicates that the new configurations are not very different or very similar to the configurations for which the neural network is trained, and the second peak is close to 0.05. Comparing Fig 8.11 with Fig 8.5 where the peak was close to 0.095, it can be concluded that the neural network training has converged over the configuration space

involved during the simulation. Fig 8.12 shows Parzen's distribution of minimum distances for the configurations during second iteration.

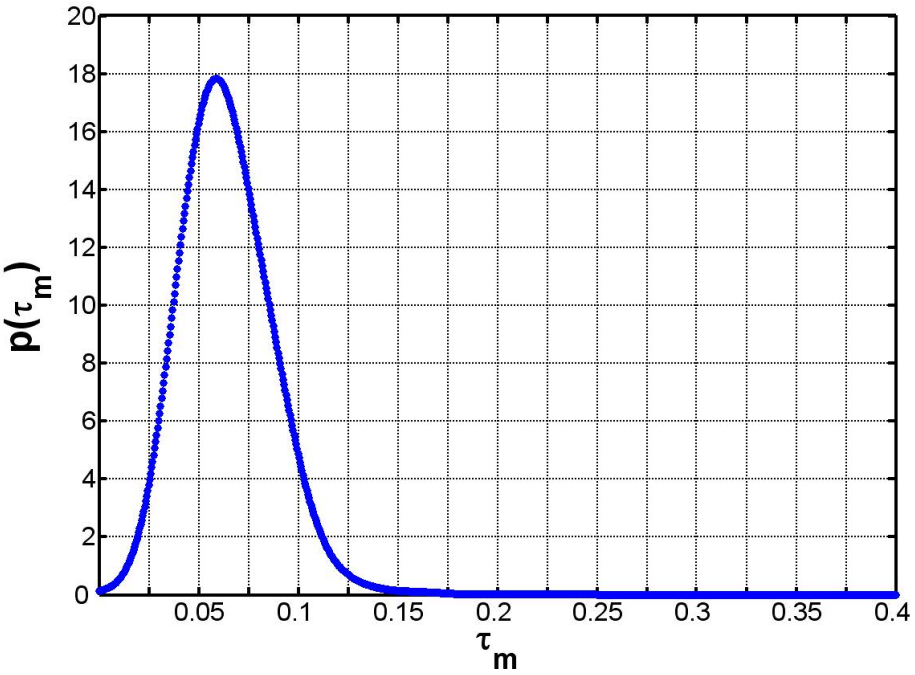


Fig 8.12 Parzen's distribution of the minimum distances for the concatenated set of configurations in the initial and first iteration.

8.4 Neural network convergence using different empirical potentials

Tersoff potential is used to model silicon workpiece during the initial step. Configurations of atoms are stored during the simulations, which are used to perform *ab initio* calculations using Gaussian 98. A neural network is then fitted to the *ab initio* energy and forces for use in MD or MC simulation. To compare the effect of using different empirical potentials as a starting point to store the configurations for Gaussian 98, two potentials proposed by Tersoff (1988, 1989) were selected. The parameters for two potentials are listed in Table 8.2.

Table 8.2 Tersoff potential parameters

	Tersoff [1] (1988)	Tersoff[2] (1988)
A (eV)	3.2647×10^{3}	1.8308×10^{3}
B (eV)	9.5373×10^{1}	4.7118×10^{2}
λ_1 ($Å^{-1}$)	3.2394	2.4799
λ_2 ($Å^{-1}$)	1.3258	1.7322
α	0.0000	0.0000
β	3.3675×10^{-1}	1.0999×10^{-6}
n	2.2956×10^{1}	7.8734×10^{-1}
c	4.8381	1.0039×10^{5}
d	2.0417	1.6217×10^{1}
h	0.0000	-5.9825×10^{-1}
λ_3 ($Å^{-1}$)	1.3258	1.7322
R ($Å$)	3.0	2.85
D ($Å$)	0.2	0.15

MD simulations were performed using the two potentials and the configurations were stored during the simulation. To compare the configurations from the two sets, minimum distances of a configuration from one set with all the configurations in the second set are computed. Fig 8.13 (a). shows a histogram of minimum distance of all configurations stored using modified Tersoff potential. Fig 8.13 (b). shows a histogram of minimum distance of configurations stored using Tersoff potential with all the configurations stored using modified Tersoff potential.

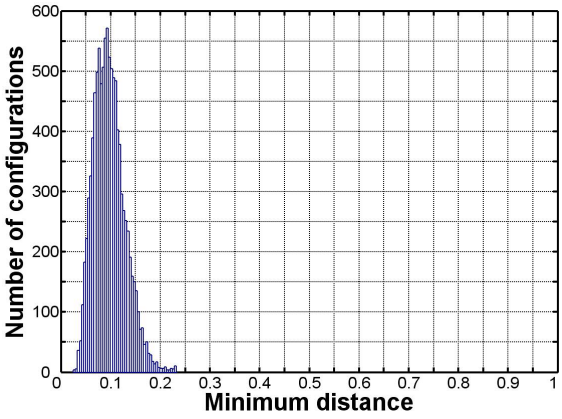
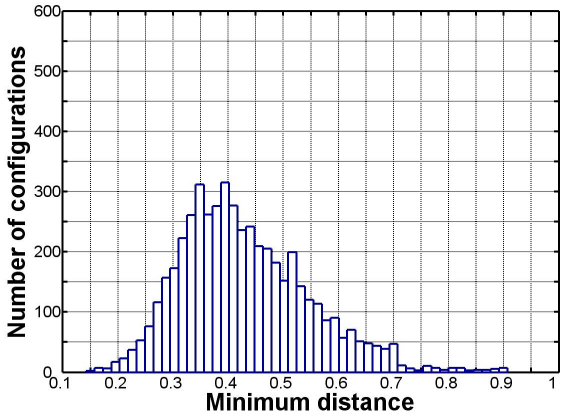


Fig 8.13 (a) Distribution of minimum distances of all configurations stored using modified Tersoff potential.



(b) Distribution of minimum distances of a configuration using Tersoff potential from configurations using modified Tersoff potential.

Larger spread of the histogram in Fig 8.13 (b) than in Fig 8.13 (a) indicates that the configurations from the two sets are very different from another.

Fig 8.14. shows a plot of the potential energy for the same set of configuration using different potential energy functions and corresponding parameters. If the values of potential energy were the same for a given configuration then the points would lie along a 45^0 line. A random distribution of points in Fig 8.14. shows that the two potentials result in different configurations.

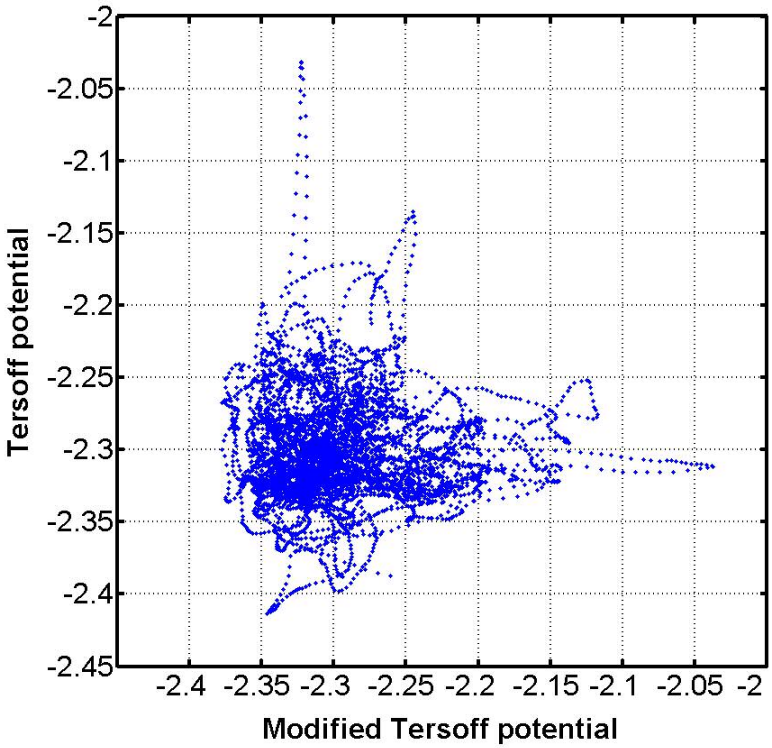


Fig 8.14 comparison of Tersoff and modified Tersoff potential energies (eV) for the same set of configurations

Next step is to compute *ab initio* energies and forces using Gaussian 98 (using Density Functional Theory) for these two sets of configurations and fit a neural network. Fig 8.15. shows a plot of neural network output for configurations trained using two sets.

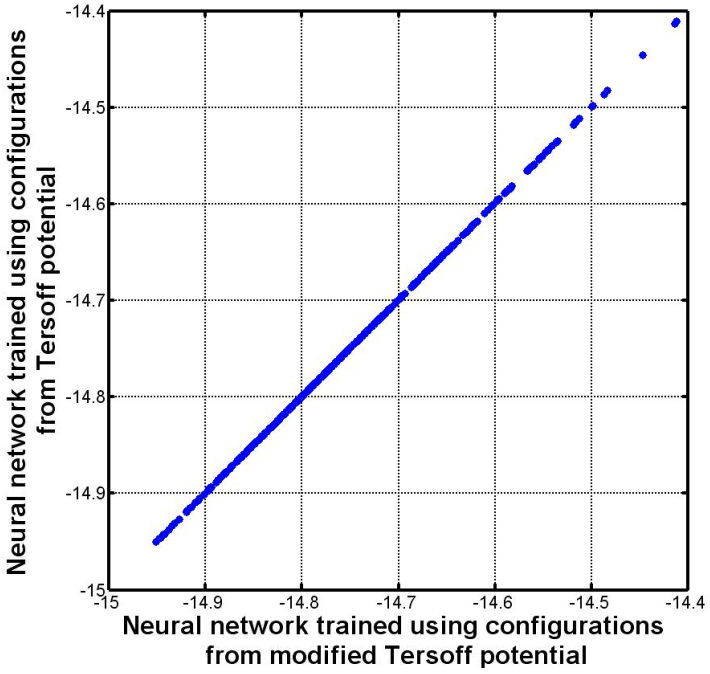


Fig 8.15 Comparison of neural network output energy (eV) for configurations from Tersoff and modified Tersoff potentials

8.5 A method to obtain a conservative force field

Ab initio calculations were performed for clusters of five silicon atoms and a neural network was trained for either the energy of the cluster or forces on the central atom in the cluster. During the MD or MC simulation this neural network is used to calculate the energy and forces on all atoms in the system. Such simulations are performed on the bulk silicon, whereas the forces on atoms are calculated using neural network trained for energy or forces in clusters of five silicon atoms. So in effect the entire work-piece is considered to be made up of separate five atoms' clusters, and the corresponding force field represents an isolated cluster of five silicon atoms and not five

silicon atoms in the bulk in which case the force field is affected not only by first neighbors but also second neighbors and so on.

If during the MD simulation, only the force on the central atom is computed for every cluster in the workpiece using the neural network then the resulting force field will not be conservative, i.e. the sum of the forces on all atoms in *X*, *Y* and *Z* directions does not add to zero. To get a conservative force field it is necessary to compute the forces on the end atoms (atoms bonded to the central atom within a five atoms' cluster) in addition to the central atom. For this purpose all workpiece atoms are grouped into two groups viz. central and end atoms.

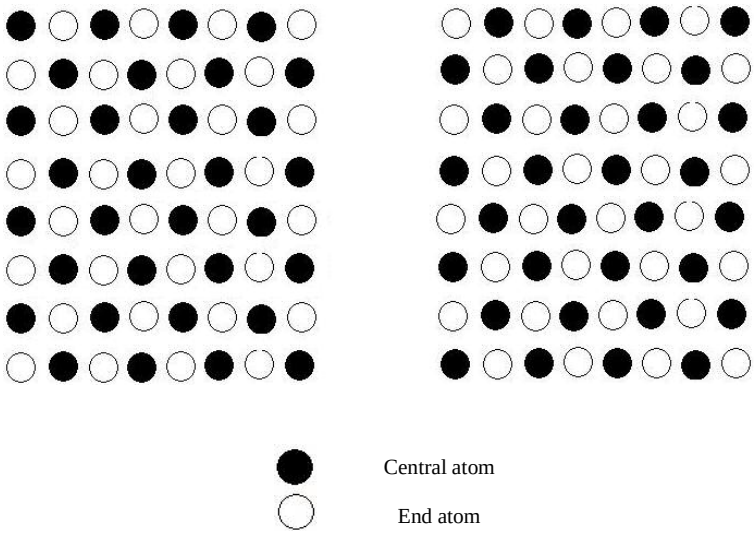


Fig 8.16 Central and end atoms grouping

During the MD simulation the force is calculated by differentiating the neural network with respect to the coordinates of the central and end atoms. For one set of central and end atoms this force field is denoted as F_1, for other set obtained by switching

the central and end atoms, the force field is denoted as F_2. It should be noted that both *F1* and *F2* produce a conservative force field.

Although *F1* or *F2* produce a conservative force field but they still represent forces in a five atoms' isolated silicon clusters and not those in a five atoms' clusters within a bulk. As a result the electron density around the end atoms and the forces on the end atoms may not be correct to model the bulk system. One approximation could be to use average force field of F_1 and F_2. This is achieved by first considering one set of central and corresponding end atoms and computing the force field denoted as F_1, and then switching the central and end atoms to obtain the force field denoted as F_2, and now the force on each atom is computed as average of F_1 and F_2. Using this approach and the force field from DFT calculations with 6-31G** basis set and B3LYP procedure for incorporating the correlation, the equilibrium bond distance was found to be 2.432 Å.

Fig 8.17 and Fig 8.18 show the variation of the total and the potential energy corresponding to the equilibrium bond distance of 2.432 Å using periodic boundary conditions.

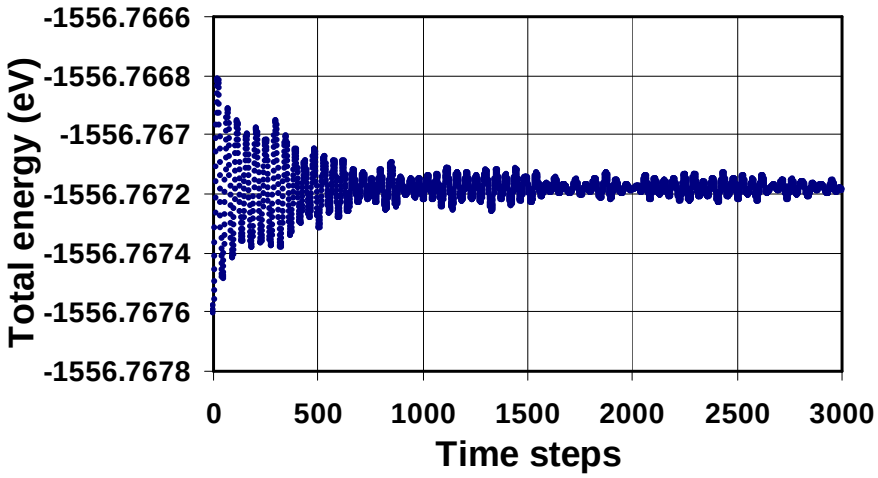


Fig 8.17 Variation of the total energy

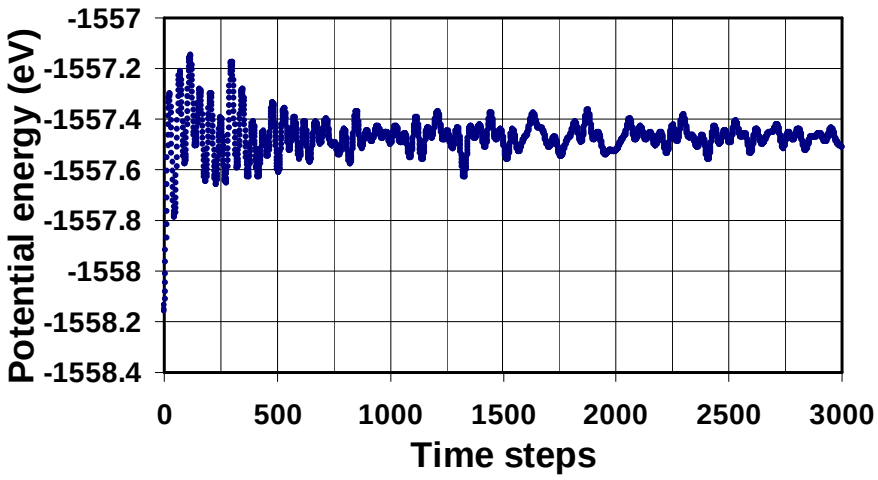


Fig 8.18 Variation of the potential energy

Fig 8.19 shows the phonon frequency plot of the four bonds of the central atom.

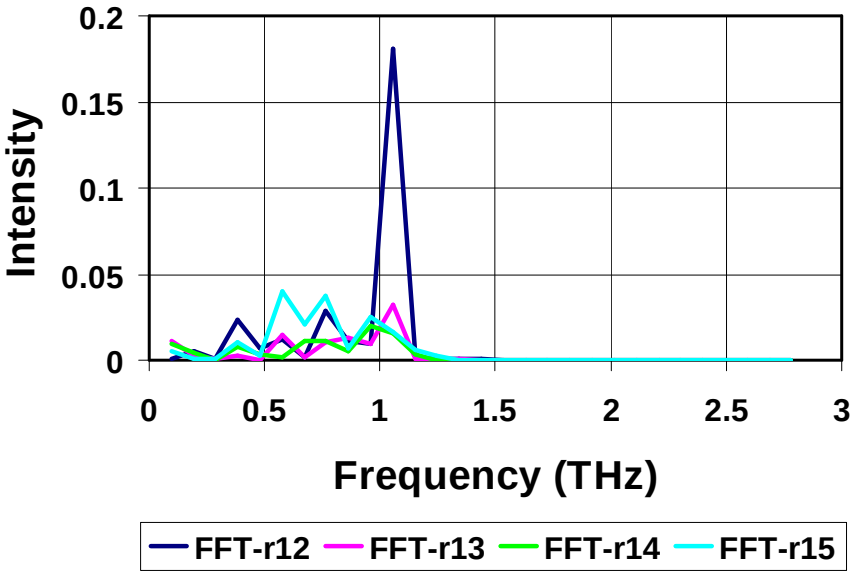


Fig 8.19 Phonon frequencies for silicon

Table 8.3 shows the comparison of the phonon frequencies obtained from the simulation and experimental values. The difference between computed and experimental frequencies is attributed to the fact that Hartree-Fock theory predicts frequencies to be 10-20% higher than experimental. Also the forces corresponds to 5-atoms' silicon clusters and not 5-atoms' within a bulk.s

Table 8.3

Experimental frequencies (THz)	Frequencies obtained from simulations by Tersoff (THz)	Frequencies obtained from simulations using NN trained for DFT energies (THz)
0.6	0.7	0.767
0.8	0.8	1.06
1.7	1.5	1.63

8.6 Neural network training for surface and bulk configurations

The size of an input vector is fixed for a given neural network, which in this case depends on the number of atoms within the cutoff radius. In bulk silicon each atom has a coordination number of four, but for the atoms on the surface the coordination number differs from the bulk. Consequently the neural network trained for the bulk silicon having nine input variables can not be used to model s structure with coordination number other four, such as the configurations of the surface atoms. To make the coordination number for surface atoms to be four, second neighbors should be included, though these second neighbors would be out of the cutoff radius for the empirical potential.

MD simulations were performed on the silicon workpiece, and the 5000 configurations for the surface atoms were stored, where every configuration has five atoms including second neighbors. The energies and force fields were computed for these configurations using Gaussian 98 and using DFT method with 6-31G** basis set and B3LYP procedure for incorporating the correlation. A neural network was trained for the data base containing bulk as well as surface configurations. The neural network specifications are given in Table 8.4 approximately 10% of the data points were used for the validation set, 10% for the testing set and remaining for the training set. To avoid overfitting Bayesian regularization and early stopping were used. Fig 8.20 shows the plot for the neural network output for the training set. The computed root mean square deviation of the neural network fit from the *ab initio* energy values was 0.0082 eV.

Table 8.4 Neural network specification

Number of layers:	2, one hidden layer and one output layer
Number of neurons:	9 for the input layer, corresponding to the number of input variables. 45 for the first hidden layer. 1 for the output layer, corresponding to the number of output variables.
Transfer function	Hyperbolic tangent function in the hidden layer. Linear function in the output layer.
Training algorithm	Levenberg-Marquardt along with Bayesian regularization.

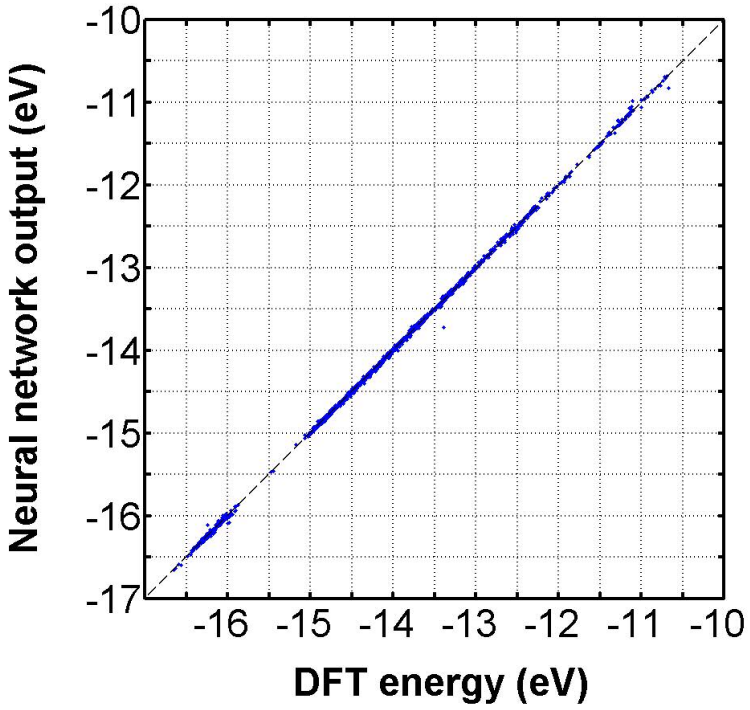


Fig 8.20 Comparison of neural network output with *ab initio* energy for the training set.

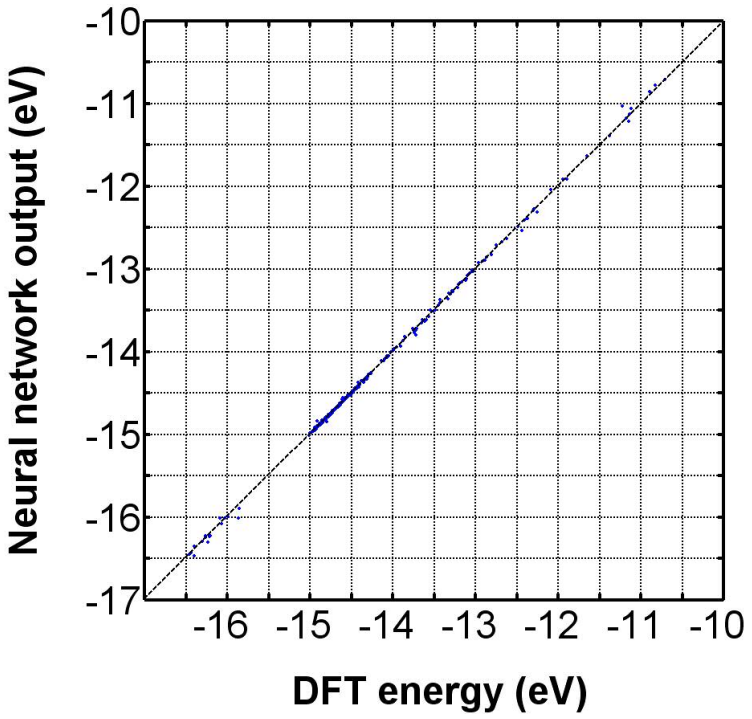


Fig 8.21Comparison of neural network output with *ab initio* energy for the validation set

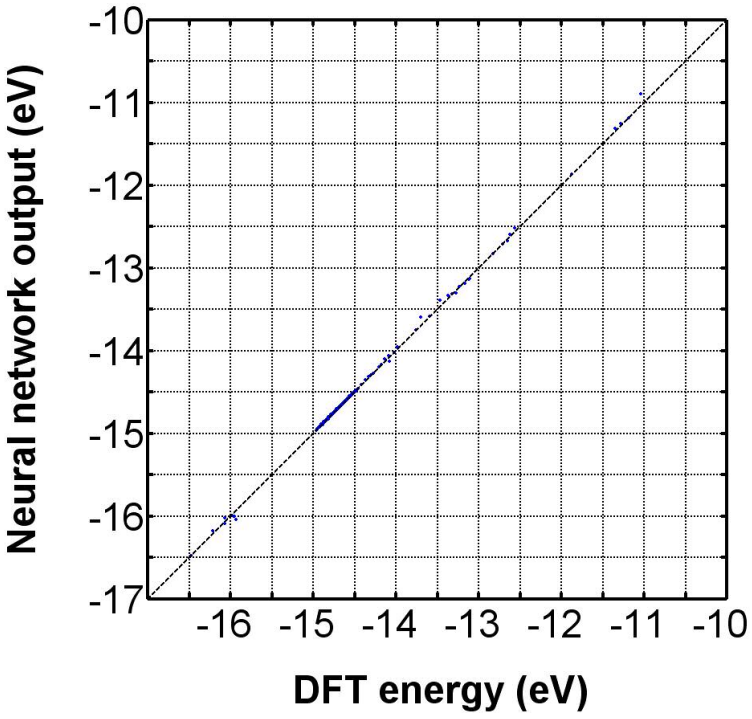


Fig 8.22Comparison of neural network output with *ab initio* energy for the testing set

8.7 Neural network training for *ab initio* energies in carbon clusters of Buckyball (C_{60}) and carbon nanotube

The energies and force fields were computed for 6,000 configurations using Gaussian 98 for density functional theory (DFT) calculations with 6-31G** basis set and B3LYP procedure for incorporating the correlation. The configurations were obtained from the simulations of buckyball (C_{60}) molecule at a temperature of 300 K, following the vibrational motions of the molecule using molecular dynamics with Tersoff potential. Although carbon atoms in a buckyball are arranged to form hexagonal and pentagonal ring structures, these rings are not planar as in the case of graphite structure. The cluster consists of four carbon atoms where three carbon atoms are bonded to one central atom. The input was specified in internal coordinates i.e. *Z-matrix* format. The variables involved in *Z-matrix* input are r_{12}, r_{13}, $r_{14,}$ θ_{312}, $\theta_{412,}$ φ_{4123}, where r_{12}, r_{13}, r_{14} are the bond distances of silicon atom #1 with atoms 2,3 and 4 respectively, θ_{312}, θ_{412} are angles of that bond 1-3 and 1-4 make with the bond 1-2 and φ_{4123} is the dihedral angle. When the trained neural network is utilized for computing energy and force fields, the same ordering of the input vector was used.

The zero of the energy is shifted so that the energy of four infinitely separated carbon atoms corresponds to zero energy. Consequently the energy of a cluster of four carbon atoms would be negative, which is the convention followed for empirical potential as well. The inputs i.e. the internal coordinates and outputs i.e. the energy are scaled so that the range for each variable in the entire database is [-1, +1], using (8.1)

The scaled database is divided into three sets: training, validation and testing set. Approximately 10% f the data points were used for the validation set, 10% for the testing

set and remaining for the training set. A multilayer feed forward neural network was used to fit the data. During the training process Bayesian regularization along with early stopping is employed to avoid overfitting.

Table 8.5 Neural network specification

Number of layers:	2, one hidden layer and one output layer
Number of neurons:	6 for the input layer, corresponding to the number of input variables. 24 for the first hidden layer. 1 for the output layer, corresponding to the number of output variables.
Transfer function	Hyperbolic tangent function in the hidden layer. Linear function in the output layer.
Training algorithm	Levenberg-Marquardt along with Bayesian regularization.

Fig 8.23 shows the plot of neural network output for the training, validation and testing sets. The computed root mean square deviation of the neural network fit from the *ab initio* energy values was 0.0054 eV.

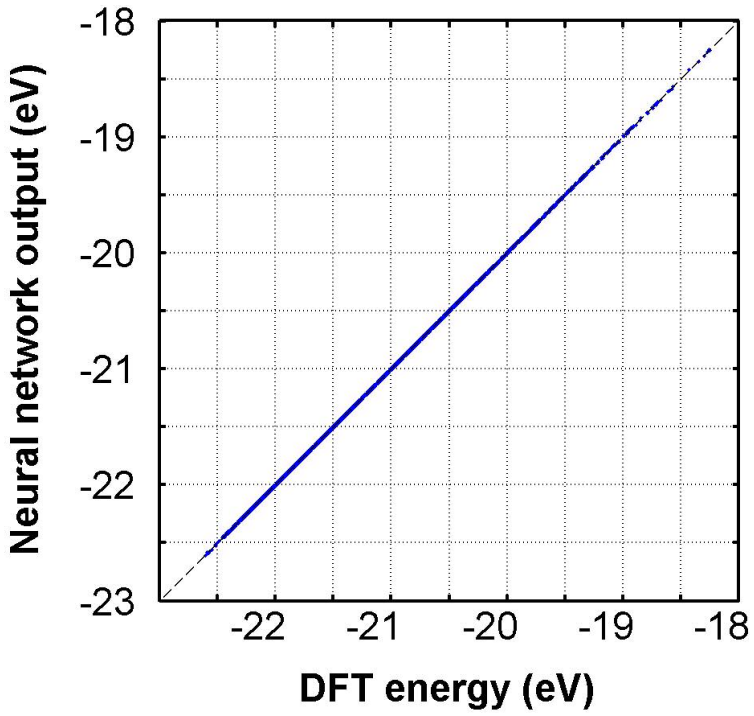


Fig 8.23 Comparison of neural network output with *ab initio* energy for the training set

CHAPTER 9

Conclusions and Future Work

An integrated approach for the development of accurate potential energy hypersurfaces based on *ab initio* electronic structure calculations and function approximation capability of multilayer neural networks has been developed, along with sampling technique based on modified *novelty detection* was developed to determine the critical regions of the configuration space. In the first step MD simulations were performed for machining of silicon using Tersoff potential. Atomic configurations of the five-atom silicon clusters that are present predominantly in the chip in front of the tool in the shear zone and within a few unit cell distances in the workpiece beneath the tool were stored during the simulation. The database of configurations obtained from the machining simulations was augmented with additional five-atom silicon configurations near the equilibrium by following the vibrational motions of the lattice. These configurations make up the initial set of configurations and serve as the starting point for the iterative process. *Ab initio* calculations were performed using density functional theory (DFT) within Gaussian 98 electronic structure program, to compute the energy and forces for these configurations. A multilayer neural network was trained with the configurations specified in internal coordinates as the input and the corresponding energies from *ab initio* calculations as the output. To avoid overfitting the underlying function, early

stopping along with Bayesian regularization was used. To reduce the time required for the neural network training, initially scaled conjugate gradient method was employed, which is faster than Levenberg-Marquardt algorithm. Once the network is reasonably trained and the improvement in the error starts reducing, then the training was stopped and restarted to employ Levenberg-Marquardt algorithm to achieve a better accuracy.

Molecular dynamics simulations were performed using the output from the trained neural network for computing the force field. Atomic configurations were stored during the simulation and compared with the initial set of configurations, for which the neural network was trained. Modified novelty detection technique was used to identify the outlier configurations i.e. the configurations that are different from those in the training set. *Ab initio* calculations were performed using density functional theory (DFT) within Gaussian 98 electronic structure program for these new configurations. These configurations were added to the initial data base of configurations, and a neural network was retrained for this new data base. This neural network was used to perform MD simulations and the configurations were stored during the simulation. Minimum distance criterion was used to check the convergence of the neural network over the configuration space involved during the simulation. The root mean squared deviations of neural network predictions from quantum mechanically computed energies was found to be on the order of 0.09 to 0.36 kJ mol^{-1} for five-atom silicon system.

The advantages of the present technique can be summarized as follows:

1. Compared with other approaches such as least squares method, the use of neural networks provides a better accuracy.

2. The neural network fitting procedure does not rely upon *ad hoc* parameterized functional forms.
3. The neural network fit is analytic and hence does not result in discontinuities in higher derivatives. The output from the neural network is differentiable with respect to the input variables, so a neural network trained for *ab initio* energies can be employed in MD simulations, and the forces could be computed by differentiating the neural network output. This property provides two advantages, first one need not have to do force calculations while performing *ab initio* electronic structure calculations, since such calculations are computationally very expensive, and secondly it would be possible to get an analytic force fields for methods for which analytical expression for forces does not exist yet, but provide very accurate results, such as MP4, which is a complete fourth order method.
4. It is also possible to fit a neural network directly for the forces.
5. Sampling technique based on modified novelty detection and MD simulation provides a way to sample only the critical regions of the configuration space.
6. The accuracy of the neural network fit can be easily improved upon by incorporating new configurations.
7. The same approach can be utilized for developing potential energy and force fields for different materials as well alloys and other chemical compounds.
8. The empirical potential employed during the first step need not be highly accurate.

Future work

In the present study, the neural network was used for fitting the energy and force fields in five-atom silicon clusters. For modeling the bulk system a periodic system could be considered for *ab initio* electronic structure calculations. For system involving different cluster sizes, such as amorphous and metastable phases, a modular neural network could be employed, which can handle input vectors of different lengths.

References

Abell, G. C., "Empirical chemical pseudopotential theory of molecular and metallic bonding," Phys. Rev. B. **31**, 6184, (1985).

Alder, B. J., and T. E. Wainwright, "Studies in Molecular Dynamics.I. General Method," J. Chem. Phys. **31**, 459-466 (1959).

Alder, B. J., and T. E., Wainwright, "Phase Transition for a Hard Sphere System," J. Chem. Phys, **27.** 1208 (1957).

Allen, M. P., and D. J. Tildesley, "Computer Simulation of Liquids," Oxford University Press, (1989).

Andersen, O. K., "Linear methods in band theory," Phys. Rev. B. **12**, 3060 (1975).

Balamane, H., Halicioglu, T. and W. A. Tiller, "Comparative study of silicon empirical interatomic potentials," Phys. Rev. B. **46**, 2250, (1992).

Balková, A., and R. J. Bartlett, "The implementation of the multireference coupled-cluster method based on the single-reference formalism," J. Chem. Phys. **96**, 3739 (1992).

Baskes, M. I., "Application of the Embedded-Atom Method to Covalent Materials: A Semiempirical Potential for Silicon," Phys. Rev. Lett. **59**, 2666, (1987)

Baskes, M. I., Nelson, J. S., and A. F. Wright, "Semiempirical modified embedded-atom potentials for silicon and germanium," Phys. Rev. B. **40**, 6085, (1989).

Bazant, M. Z., Kaxiras E., and J. F. Justo, “Environment-dependent interatomic potential for bulk silicon,” Phys. Rev. B. **56,** 8542, (1997)

Becke, A. D. “Density-functional thermochemistry. IV. A new dynamical correlation functional and implications for exact-exchange mixing,” J. Chem. Phys. **104**, 1040 (1996).

Becke, A. D., "A new mixing of Hartree-Fock and local density-functional theories," J. Chem. Phys. **98**, 1372-1377 (1993).

Becke, A. D., “Density-functional exchange-energy approximation with correct asymptotic behavior,” Phys. Rev. A. **38**, 3098 (1988).

Binkley, J. S., Pople, J. A. and Hehre, W. J., “Self-consistent molecular orbital methods. 21. Small split-valence basis sets for first-row elements,” J. Am. Chem. Soc. **102**, 939 (1980).

Binning Jr., R. C. and Curtiss, L. A., “Compact contracted basis sets for third-row atoms: Ga-Kr," J. Comp. Chem. **11**, 1206 (1990).

Bishop, C. M., “Neural networks for pattern recognition,” New York: Oxford University Press Inc, (1995).

Bishop, C. M., “Novelty detection and neural network validation,” Proceedings of IEEE Vision, Image and Signal Processing, **141**, 217-222, (1994).

Biswas, R. D., R. Hamann, “New classical models for silicon structural energies,” Phys. Rev. B **36.** 6434 (1987).

Biswas, R., and D. R. Hamman, “Interatomic potentials for silicon structural energies,” Phys. Rev. Lett. **55**, 2001, (1985).

Bolding, B. C., and Anderson, H. C., "Interatomic potential for silicon clusters, crystals, and surfaces," Phys. Rev. B. **41.** 10568, (1990).

Born, M., and Meyer, M. G., Handbuch der Physik (Berlin), (1933).

Boys, S. F., Proc. R. Soc. A**200,** (1950).

Bratley, P., Fox, B. L., and L. E. Schrage. A Guide to Simulation. Springer-Verlag, New York, (1987).

Brenner, D. W., and B. J. Garrison, "Dissociative valence force field potential for silicon," Phys. Rev. B. **34,**1304, (1985).

Brenner, D. W., "Empirical potential for hydrocarbons for use in simulating the chemical vapor deposition of diamond films," Phys. Rev. B. **42**, 9458, (1990).

Brenner, D. W., "Erratum: Empirical potential for hydrocarbons for use in simulating the chemical vapor deposition of diamond films," Phys. Rev. B. **46**, 1948, (1992).

Burke, K. Perdew, J. P. and Y. Wang, in Electronic Density Functional Theory: Recent Progress and New Directions, Ed. Dobson, J. F. Vignale, G. and Das, M. P. (Plenum, 1998).

Calhoun, A. and D. J. Doren, "Quantum Effects in Elementary Surface Reactions: CO Diffusion on Ni(111)," J. Phys. Chem. **97**, 2251 (1993).

Car, R, and M. Parrinello, "Unified approach for molecular dynamics and density functional theory," Phys. Rev. Lett. **55,** 2471, (1985).

Carlson, A.E in Solid State Physics: Advances in Research and Applications, edited by H.Ehrenreich and D.Turnbul (Academic,NewYork), **43,** 1 (1990).

Carlsson, A. E. Fedders, P. A., and C. W. Myles, "Generalized embedded-atom format for semiconductors," Phys. Rev. B. **41**, 1247, (1990).

Carlsson, A. E. Gelatt, C., and Ehrenreich, H., Phil. Mag. A. **41**, (1980).

Carlsson, A. E., "Unified description of constant-volume and bond-breaking pair potentials in metals," Phys. Rev. B. **32**, 4866, (1985).

Carlsson, A. E., and Ashcroft, N. W., "Pair potentials from band theory: Application to vacancy-formation energies," Phys. Rev. B. **27**, 2101, (1983).

Carlsson, A. E., Springer Proc in Physics, **48**, 257, (1990).

Chang, K.J., Cohen. M.L, "Structural and electronic properties of the high-pressure hexagonal phases of Si ," ibid, **30** , 5376, (1984).

Chen, D and M. Hagan, "Optimal Use of Regularization and Cross-Validation in Neural Network Modelinmg", Proceedings of the 1999 International Joint Conference on Neural Networks, Wahsington, **2**, 1275 (1999).

Chen, D., and M. Hagan, "Optimal use of regularization and cross-validation in neural network modeling," IJCNN, **2,** 1275-1280, (1999).

Ciccotti, G. Frenkel, D., and I. R. McDonald, "Simulation of Liquids and Solids," Elsevier Science, (1987).

Clark, T., Chandrasekhar, J., Spitznagel, G. W. and Schleyer, P. v. R., "Efficient diffuse function-augmented basis sets for anion calculations. III. The 3-21+G basis set for first-row elements, Li-F," J. Comp. Chem. **4**, 294 (1983).

Collins, J. B., Schleyer, P. v. R., Binkley, J. S. and J. A. Pople, "Self-consistent molecular orbital methods. XVII. Geometries and binding energies of second-row molecules. A comparison of three basis sets," J. Chem. Phys. **64**, 5142 (1976).

Curtiss, L. A., McGrath, M. P., Blaudeau, J.-P., Davis, N. E., Binning Jr., R. C. and L. Radom, “Extension of Gaussian-2 theory to molecules containing third-row atoms Ga–Kr,” J. Chem. Phys. **103**, 6104 (1995).

Cybenko, G., “Approximation by Superpositions of a Sigmoidal Function,” Math. Control Signals Systems, **2**, 303–314 (1989).

Dan, Foresse, F., and M., Hagan, “Gauss-Newton approximation to Bayesian learning,” IEEE Conference Neural Networks, **3,** 1930-1935, (1997).

Demuth, H. B. and M. Beale, "Users' Guide for the Neural Network Toolbox for MATLAB, ver. 4.0, (The Mathworks, Natick, MA, 2000).

Deuflhard, P. Hermans, J. Leimkuhler, B. Mark, A. E. Reich, S., and R. D. Skeel, “Computational Molecular Dynamics: Challenges, Methods, Ideas,” Proceedings of the 2nd International Symposium on Algorithms for Macromolecular Modelling, Berlin, May 21-24, (1997).

Ditchfield, R., Hehre, W. J. and Pople, J. A. “Self-Consistent Molecular-Orbital Methods. IX. An Extended Gaussian-Type Basis for Molecular-Orbital Studies of Organic Molecules,” J. Chem. Phys. **54**, 724 (1971).

Dobbs, K. D. and Hehre, W. J., “Molecular orbital theory of the properties of inorganic and organometallic compounds 4. Extended basis sets for third-and fourth-row, main-group elements,” J. Comp. Chem. **7**, 359 (1986).

Dobbs, K. D. and Hehre, W. J., “Molecular orbital theory of the properties of inorganic and organometallic compounds 5. Extended basis sets for first-row transition metals,” J. Comp. Chem. **8**, 861 (1987).

Dobbs, K. D. and Hehre, W. J., "Molecular orbital theory of the properties of inorganic and organometallic compounds. 6. Extended basis sets for second-row transition metals," J. Comp. Chem. **8**, 880 (1987).

Dobbs, K. D., and D. J. J. Doren, "Dynamics of molecular surface diffusion: Origins and consequences of long jumps," J. Chem. Phys. **97**, 3722 (1992).

Dobbs, K. D., and D. J. J. Doren, "Dynamics of molecular surface diffusion: Energy distributions and rotation–translation coupling," J. Chem. Phys. **99**, 10041 (1993).

Doren, D. J. and J. C. Tully, "Dynamics of precursor-mediated chemisorption," J. Chem. Phys. **94**, 8428 (1991).

Doren, D. J. and J. C. Tully, "Precursor-mediated adsorption and desorption: a theoretical analysis," Langmuir, **4**, 256 (1988).

Dovesi, R. Saunders, V. R., and C. Roetti, CRYSTAL 92 User Manual, University of Turin, Italy, and SERC Daresbury Laboratory, UK; August 17 (1992).

Dunning Jr., T. H. and P. J. Hay, in Modern Theoretical Chemistry, Ed. H. F. Schaefer III, Vol. 3 (Plenum, New York) (1976).

Dyson, A. J., and P. V. Smith, "Extension of the Brenner empirical interatomic potential to C-Si-H systems," Surface science, **355,** 140-150, (1996).

Ercolessi, F., and Adams, J. B., "Developing empirical models from *ab initio* calculations - application to Al," Europhysics letters, **26**, 583, (1994).

Ercolessi, F., and J. Adams, "Interatomic potentials from first principles," Materials theory and modeling symposium, **291,** 31-36, (1993).

Ewald, P., "Die Berechnung optischer und elektrostatischer gitterpotentiale," Ann Phys **64**, 253-87 (1921).

Fermi, E., Z. Physik. **48**, 73 (1928).

Fock, V., ibid. **62**, 795 (1930).

Fock, V., Z. Phys. **61**, 126 (1930).

Foresee, F. D. and M. Hagan, "Gauss-Newton Approximation to Bayesian Regularization", Proceedings of the 1997 International Conference on Neural Networks, Houston , Texas (1997).

Foresman, J. B., and E. Frisch, "Exploring Chemistry with Electronic Structure Methods," Gaussian Inc, (1993).

Frenkel, D., and B. Smit, "Understanding Molecular Simulation," Academic Press, (2001).

Frisch, M. J., Head-Gordon, M. and J. A. Pople, "Semi-direct algorithms for the MP2 energy and gradient," Chem. Phys. Lett. **166**, 281 (1990).

Frisch, M. J., Head-Gordon, M.and J. A. Pople, "A direct MP2 gradient method," Chem. Phys. Lett. **166**, 275 (1990).

Frisch, M. J., Pople, J. A. and J. S. Binkley, "Self-consistent molecular orbital methods 25. Supplementary functions for Gaussian basis sets," J. Chem. Phys. **80**, 3265 (1984).

Frosini, A. Gori, M., and Priami, P., "A neural network based model for paper currency recognition and verification," IEEE Transactions on Neural Networks, **7(6)**, 1482-1490, (1996).

Garifalco, L. A., and V. G. Weizer, "Application of the Morse potential function to cubic meals," Phys. Rev. **114**, 687, (1959).

Gaussian 98 (Revision A.1), M. J. Frisch, G. W. Trucks, H. B.Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, V. G. Zakrzewski, J. A. Montgomery, Jr., R. E. Stratmann, J. C. Burant, S. Dapprich, J. M. Millam, A. D. Daniels, K. N. Kudin, M. C. Strain, O. Farkas, J. Tomasi, V. Barone, M. Cossi, R. Cammi, B. Mennucci, C. Pomelli, C. Adamo, S. Clifford, J. Ochterski, G. A. Petersson, P. Y. Ayala, Q. Cui, K. Morokuma, P. Salvador, J. J. Dannenberg, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. Cioslowski, J. V. Ortiz, A. G. Baboul, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. Gomperts, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, J. L. Andres, C. Gonzalez, M. Head-Gordon, E. S. Replogle, and J. A. Pople, Gaussian, Inc., Pittsburgh PA, (2001).

Geman S., Bienenstock E., R. Doursat, “Neural networks and the bias/variance dilemma”,Neural Computation, **4**¸1-58 (1992).

Gibson, J. B., Goland A. N., Milgram M., and G. H. Vineyard, “Dynamics of Radiation Damage,” Phys. Rev. **120**, 1229 (1960).

Goldstein, H., "Classical Mechanics," Addison-Wesley, Reading, MA. (1965).

Gordon, M. S., ‘The isomers of silacyclopropane,” Chem. Phys. Lett. **76**, 163 (1980).

Gordon, M. S., Binkley, J. S., Pople, J. A., Pietro, W. J. and W. J. Hehre, “Self-consistent molecular orbital methods. 22. small split-valence basis sets for second-row elements,” J. Am. Chem. Soc. **104**, 2797 (1982).

Gunnarsson, O., Lundqvist, B. I. and S. Lundqvist, Solid State Commun.**11**, 149 (1972).

Hagan, M. T. and M. Menhaj, “Training feedforward neural networks with the Marqurdt algorithm,” IEEE Transactions on Neural Networks, **5**, 989 (1994).

Hagan, M. T., Demuth, H. B.and M. Beale, "Neural Network Design", PWS Publishing Company, Boston, MA, (1996).

Hagan, M., “Backpropagation” Chapter 5 of the User's Guide for the Neural Network Toolbox for MATLAB, ver. 3, H. Demuth and M. Beale, Natick, MA: The Mathworks Inc., (1998).

Haile, J. M., “Molecular Dynamics Simulation: elementary methods,” New York : Wiley, (1992).

Handy, N. C. and Schaefer, III, H. F., “On the evaluation of analytic energy derivatives for correlated wave functions,” J. Chem. Phys. **81**, 5031 (1984).

Hansen, J.P. and L. Verlet, “Phase Transitions of the Lennard-Jones System,” Phys. Rev. **184**, 151 (1969).

Hariharan, P. C. and J. A. Pople, “Equilibrium Geometries by Single Determinant Molecular Orbital Theory,” Mol. Phys. **27**, 209 (1974).

Hariharan, P. C. and J. A. Pople, “The Influence of Polarization Functions on Molecular Orbital Hydrogenation Energies,” Theo. Chim. Acta **28**, 213 (1973).

Hartree, D. R., Proc. Camb.Phil. Soc. **24**, 89 (1928).

Hay, P. J., “Gaussian basis sets for molecular calculations. The representation of 3*d* orbitals in transition-metal atoms,” J. Chem. Phys. **66**, 4377 (1977).

Haykin, S. "Neural Networks: A Comprehensive Foundation" (Maxwell MacMillan International).

Head-Gordon, M. and Head-Gordon, T., “Analytic MP2 frequencies without fifth-order storage. Theory and application to bifurcated hydrogen bonds in the water hexamer ,” Chem. Phys. Lett., **220**, 122 (1994).

Head-Gordon, M., Pople, J. A. and Frisch, M. J. “MP2 energy evaluation by direct methods,” Chem. Phys. Lett. **153**, 503 (1988).

Hebb, D. O., “The organization of behavior,” New York: Wiley, (1949).

Hehre, W. J. Radom, L. Schleyer, P. and J. A. Pople, “Ab initio molecular orbital theory,” John Wiley & Sons, New York (1986).

Hehre, W. J., Ditchfield, R. and Pople, J. A. “Self Consistent Molecular Orbital Methods. XII. Further Extensions of Gaussian—Type Basis Sets for Use in Molecular Orbital Studies of Organic Molecules,” J. Chem. Phys. **56**, 2257 (1972).

Hehre, W. J., Stewart, R. F. and Pople, J. A., “Self-Consistent Molecular-Orbital Methods. I. Use of Gaussian Expansions of Slater-Type Atomic Orbitals,” J. Chem. Phys. **51**, 2657 (1969).

Hertz, J. Krogh, A. and R. G. Palmer, Introduction to the Theory of Neural Computation (Addison-Wesley, Reading, 1996).

Hickinbotham, S. J., and Austin, J., “Neural networks for novelty detection in airframe strain data,” Proceedings of the IJCNN, **6**, 375-380, (2000).

Hohenberg, P. and Kohn, W., “Inhomogeneous electron gas,”Phys. Rev. **136**, B864 (1964).

Ischtwan, J., and M. A. Collins, “Molecular potential energy surfaces by interpolation,” J. Chem. Phys. **100**, 8080 (1994).

Jordan, M. J. T. Thompson, K. C., and M. A. Collins, “Convergence of molecular potential energy surfaces by interpolation: Application to the $OH+H_2 \rightarrow H_2O+H$ reaction,” J. Chem. Phys. **102**, 5647, (1995).

Justo J. F., Bazant M. Z., Kaxiras E., Bulatov V. V., and S. Yip, “Interatomic potential for silicon defects and disordered phases,” Phys. Rev. B. **58,** 2539 (1998).

Justo, J. F., Bazant, M. Z., Kaxiras, E., Bulatov, V. V., and S. Yip, Materials Research Society Proc. **469**, (1997).

Kane, E.O., “Phonon spectra of diamond and zinc-blende semiconductors,” Phys Rev B. **31**, 7865, (1985).

Karplus, M., and G. A. Petsko, Molecular Dynamics Simulations in Biology, Nature, **347**, 631-639 (1990).

Kaxiras, E., “Review of atomistic simulations of surface diffusion and growth on semiconductors,” Comp.Mater.Sci., **6**,158(1996).

Keating, P. N., “Effect of Invariance Requirements on the Elastic Strain Energy of Crystals with Application to the Diamond Structure,” Phys. Rev. **145**, 637, (1966).

Kendall, R. A. Dunning Jr., T. H. and Harrison, R. J., “Electron affinities of the first-row atoms revisited. Systematic basis sets and wave functions,” J. Chem. Phys. **96**, 6796 (1992).

Kirkpatrick, S., Gelatt, C.D. and M.P. Vecchi, “Optimization by simulated annealing,” Science, **220**, 671, (1983).

Koelling, D. D., and G. O. Arbman, J. Phys. F. **5**, 2041 (1975).

Kohn, W. and Sham, L. J., “Self-Consistent Equations Including Exchange and Correlation Effects,” Phys Rev. **140**, A1133 (1965).

Kohn, W., and N. Rostoker, “Solution of the Schrödinger Equation in Periodic Lattices with an Application to Metallic Lithium,” Phys. Rev. **94**, 1111 (1954).

Komanduri, R., and L. M. Raff, “Molecular dynamics simulation of machining,” in New developments in the finishing of advanced materials, Proceedings of the US-Japan symposium, 153, (1999).

Komornicki, A., and G. Fitzgerald, Chem. Phys. **98**, 1399 (1993).

Korringa, J., Physica, **13**, 392 (1947).

Krishnan, R. and Pople, J. A., “Approximate fourth-order perturbation theory of the electron correlation energy,” Int. J. Quant. Chem. **14**, 91 (1978).

Krishnan, R., Binkley, J. S., Seeger, R. and Pople, J. A. “Self-consistent molecular orbital methods. XX. A basis set for correlated wave functions,” J. Chem. Phys. **72**, 650 (1980).

Labanowski, J., and J. Andzelm, editors, Density Functional Methods in Chemistry, Springer-Verlag, New York, (1991).

Leach, A., “Molecular Modeling: Principles and Applications,” Prentice Hall, (2001).

LeCun, Y. Bottou, L. Orr, G. B. and MÄuller, K. R.in Neural Networks: Tricks of trade, G. B. Orr and K. R. Muller (eds), Springer Verlag, Heidelberg, (1998).

Li, X. P. Nunes, R. W., and D. Vanderbilt, “Density-matrix electronic-structure method with linear system-size scaling,” Phys. Rev. B. **47**, 10891 (1993).

Madelung, E., Phys. Z. **19,** 524-32 (1918)

Maisuradze, G. C. Thompson, D. L. Wagner, A. F., and M. Minkoff, J. Chem. Phys. **119**, 10002, (2003).

Makarov, D. E., and H. Metiu, “Fitting potential-energy surfaces: A search in the function space by directed genetic programming,” J. Chem. Phys. **108**, 590, (1998).

Martinez, D., “Neural tree density estimation for novelty detection,” IEEE Transctions on Neural Networks, **9(2)**, 330-338, (1998).

Marx, D., and J. Hutter, “*ab initio* molecular dynamics: Theory and implementation,” Modern methods and algorithms of quantum chemistry, John von Neumann Institute for Computing, Julich, NIC Series, **1**, 301-449, (2000).

Masters, T. “Practical Neural Network Recipes,” John Wiley, New York, (1993).

Mauri, F. Galli, G., and R Car, “Orbital formulation for electronic-structure calculations with linear system-size scaling,” Phys. Rev. B. **47**, 9973 (1993).

McCulloch, W. and Pitts, W., “A logical calculus of the ideas immanent in nervous activity,” Bulletin of Mathematical Biophysics, **5** 115-133, (1943).

McGrath, M. P. and L. Radom, “Extension of Gaussian-1 (G1) theory to bromine-containing molecules,” J. Chem. Phys. **94**, 511 (1991).

McLean, A. D., and G. S. Chandler, “Contracted Gaussian basis sets for molecular calculations. I. Second row atoms, *Z*=11–18,” J. Chem. Phys. **72**, 5639 (1980).

Minsky, M. and Papert, S., “Perceptrons,” Cambridge, MA: MIT Press, (1969).

Mistriotis, D. Flytzanis, A., and S.C. Farantos, “Potential model for silicon clusters,” Phys.Rev. B. **39**,1212 (1989).

Moller, C. and Plesset, M. S., “Note on an Approximation Treatment for Many-Electron Systems,” Phys. Rev. **46**, 618 (1934).

Morse, P. M., “Diatomic molecules according to the wave mechanics II vibrational levels,” Phys. Rev. **34**, 57, (1929).

Nairac, A. Corbett-Clark, T. Ripley, R. Townsend, N., and L. Tarassenko, "Choosing an appropriate model for novelty detection," Proceedings of the 5th International Conference on Artificial Neural Networks, 117-122, (1997).

Pai, S. and D. Doren, "Dynamics of molecular surface diffusion: energy transfer between adsorbate modes," Surf. Sci. **291**, 185 (1993).

Parr, R. G. and W. Yang, Density-functional theory of atoms and molecules (Oxford Univ. Press, Oxford, 1989).

Payne, M. C. and J. D. Joannopoulos, Allan, D. C. Teter, M. P., and D. H. Vanderbilt, "Molecular Dynamics and *ab initio* Total Energy Calculations," Phys. Rev. Lett. **56**, 2656, (1986). (Comment on Unified approach for molecular dynamics and density functional theory by Car and Parrinello),

Perdew, J. P. Burke, K. and Y. Wang, Generalized gradient approximation for the exchange-correlation hole of a many-electron system," Phys. Rev. B. **54**, 16533 (1996).

Perdew, J. P., "Density-functional approximation for the correlation energy of the inhomogeneous electron gas," Phys. Rev. B. **33**, 8822 (1986).

Petersson, G. A. and Al-Laham, M. A., "complete basis set model chemistry. II. Open-shell systems and the total energies of the first-row atoms," J. Chem. Phys. **94**, 6081 (1991).

Petersson, G. A., Bennett, A., Tensfeldt, T. G., Al-Laham, M. A., Shirley, W. A. and J. Mantzaris, "A complete basis set model chemistry. I. The total energies of closed-shell atoms and hydrides of the first-row elements," J. Chem. Phys. **89**, 2193 (1988).

Pettifor D. G., “New many-body potential for the bond order,” Phys. Rev. Lett. **63**, 2480, (1989).

Pettifor, D. G., Springer Proc in Physics, **48**, 64, (1990).

Pietro, W. J., Francl, M. M., Hehre, W. J., Defrees, D. J., Pople, J. A. and J. S. Binkley, J. Am. Chem. Soc. **104**, 5039 (1982).

Pines, D., Solid State Physics **1**, 367 (1951).

Pople, J. A., Binkley, J. S. and Seeger, R. “Theoretical models incorporating electron correlation,” Int. J. Quant. Chem. Symp. **10**, 1 (1976).

Pople, J. A., Krishnan, R., Schlegel, H. B. and Binkley, J. S. Int. J. Quant. Chem. Symp. **13**, 325 (1979).

Pople, J. A., Seeger, R. and Krishnan, R. Int. J. Quant. Chem. Symp. **11**, 149 (1977).

Pukrittayakamee, A. "Novelty detection for function approximation,” PhD thesis, Oklahoma state university, (2001).

Raff, L. M. and D. L.Thompson, "The Classical Trajectory Approach to Reactive Scattering" in Theory of Chemical Reaction Dynamics, M. Baer, Ed. Vol. III, 1 (CRC Press, Boca Raton, FL) (1985).

Raghavachari, K., Pople, J. A., Replogle, E. S. and M. Head-Gordon, “Fifth order Moeller-Plesset perturbation theory: comparison of existing correlation methods and implementation of new methods correct to fifth order,” J. Phys. Chem. **94**, 5579 (1990).

Raghavachari, K.and Trucks, G. W., “Highly correlated systems. Excitation energies of first row transition metals Sc–Cu,” J. Chem. Phys. **91**, 1062 (1989).

Rahaman, A., and Raff, L. M., “Trajectory investigations of the dissociation dynamics of vinyl bromide on an *ab initio* potential-energy surface,” J. Phys. Chem. A. **105**, 2156, (2001).

Rahman A., “Correlations in the Motion of Atoms in Liquid Argon,” Phys. Rev. **136**, A405 (1964).

Rahman, A., and F. H. Stillinger, Molecular Dynamics Study of Liquid Water, J. Chem. Phys. **55**, 3336-3359 (1971).

Rapaport, D. C., “The Art of Molecular Dynamics Simulation,” Cambridge University Press, (2004).

Rentsch, R., and Inasaki, I.., "Molecular Dynamics Simulation for Abrasive Processes," Annals of CIRP, **43**, 327, (1994).

Rosenblatt, F., “The Perceptron: A probabilistic model for information storage and organization in the brain,” Psychological Review, **65,** 386-408, (1958).

Saebo, S. and Almlof, J., “Avoiding the integral storage bottleneck in LCAO calculations of electron correlation,” Chem. Phys. Lett. **154**, 83 (1989).

Salahub, D. R. and M. C. Zerner, The Challenge of d and f Electrons, (ACS, Washington, D.C., (1989).

Sarle, W. S., "Stopped Training and Other Remedies for Overfitting", Proceedings of the 27th Symposium on the Interface, (1995).

Schlick, T., “Molecular Modeling and Simulation,” Springer-Verlag, (2002).

Schultz, R., and M. Hagan, “On-Line Least Squares Learning for the Underdetermined Case,” International Joint Conference on Neural Networks, Washington, Paper No. 515, (1999).

Schultz, R., and M. Hagan, "Training Multiloop Networks," International Joint Conference on Neural Networks, Washington, Paper No. 514. (1999).

Slater, J. C., "A Simplification of the Hartree-Fock Method," Phys. Rev. **81**, 385 (1951).

Slater, J. C., "Damped Electron Waves in Crystals," Phys. Rev. **51**, 840 (1937).

Slater, J. C., "Wave Functions in a Periodic Potential," Phys. Rev. **51**, 846 (1937).

Slater, J. C., Quantum Theory of Molecules and Solids Vol. 4, McGraw-Hill, New York (1974).

Specht, D. F. "Probabilistic Neural Networks and the Polynomial Adaline as Complementary Techniques for Classification," IEEE Transactions on Neural Networks, **1**, 111-121, (1990).

Specht, D.F., "Probabilistic Neural Networks," Neural Networks, **3**, 109-118, (1990).

Stilinger, F.H., Weber, T.A., "Computer simulation of local order in condensed phases of silicon," Phys.Rev. B. **31**, 5262 (1985).

Tersoff, J., "Empirical potential for carbon, with applications to amorphous carbon," Phys. Rev. Lett. **61**, 2879 (1988).

Tersoff, J., "Modeling solid state chemistry: Interatomic potentials for multicomponent systems," Phys. Rev. B. **39**, 5566 (1989).

Tersoff, J., "New empirical approach for the structure and energy of covalent systems," Phys. Rev. B. **37**, 6991 (1988).

Tersoff, J., "New empirical model for the structural properties of silicon," Phys. Rev. Lett. **56**, 632, (1986).

Thomas, L. H., Proc. Camb. Phil. Soc. **23**, 542 (1926).

Torii, M., and M. Hagan, “Stability of Steepest Descent with Momentum for Quadratic Functions,” IEEE Transactions on Neural Networks, **13**, 3, 752 -756, (2002).

Torrens, I. M., “Interatomic potentials,” Academic Press, New York, (1972).

Trucks, G. W., Salter, E. A., Sosa, C. and R. J. Bartlett, “Theory and implementation of the MBPT density matrix. An application to one-electron properties,” Chem. Phys. Lett. **147**, 359 (1988).

Trucks, G. W., Watts, J. D., Salter, E. A. and R. J. Bartlett, “Analytical MBPT(4) gradients,” Chem. Phys. Lett. **153**, 490 (1988).

Verlet L., “Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules,” Phys. Rev. **159**, 98 (1967).

Verlet L., “Computer "Experiments" on Classical Fluids. II. Equilibrium Correlation Functions,” Phys. Rev. **165**, 201 (1967).

Vitek, V., “Pair potentials in atomistic computer simulations,” MRS Bulletin, 20, (1996).

von Barth, U., and L. J. Hedin, Phys. C **5**, 1629 (1972).

Voter, A. F., “Interatomic potentials for atomic simulatons,” MRS Bulletin, 17, (1996).

Wachters, A. J. H., “Gaussian Basis Set for Molecular Wavefunctions Containing Third-Row Atoms,” J. Chem. Phys. **52**, 1033 (1970).

Wasserman, P.D. “Advanced Methods in Neural Networks,” Van Nostrand Reinhold, New York, (1993).

Webros, P.J., “Beyond regression: Net tools for prediction and analysis in the behavioral sciences”, PhD thesis, Harvard University (1974).

Widrow, B. and Hoff, M. E., “Adaptive switching circuits,” IRE WESCON Convention Record, New York: IRE part **4**, 96-104, (1960).

Woon D. E. and T. H. Dunning Jr., “Gaussian basis sets for use in correlated molecular calculations. III. The atoms aluminum through argon,” J. Chem. Phys. **98**, 1358 (1993).

Yin, M.T. and Cohen, M.L. “Microscopic Theory of the Phase Transformation and Lattice Dynamics of Si,” Phys. Rev. Lett. **45**, 1004, (1980).

Yin, M.T. and Cohen, M.L., “Structural theory of graphite and graphitic silicon,” Phys. Rev. B. **29**, 6996, (1984).

Yin, M.T. and Cohen, M.L., “Theory of static structural properties, crystal stability, and phase transformations: Application to Si and Ge,” Phys. Rev. B. **26,** 5668, (1982).

Appendix

Steps involved in developing neural network trained *ab initio* potential energy hypersurfaces

1. Perform molecular dynamics/ Monte Carlo simulations of a material under consideration using an appropriate potential for modeling interatomic interactions. Calculate the number of atoms within a cutoff radius for an atom in the material, and store coordinates of atoms forming a cluster. The configurations that are generated in an ensemble of trajectories using an empirical potential comprise the subset of configuration space that has to be sampled.

   Following example shows a cluster of 5 silicon atoms, Si_1 is the central atom to which atoms Si_2, Si_3, Si_4, and Si_5 are bonded.

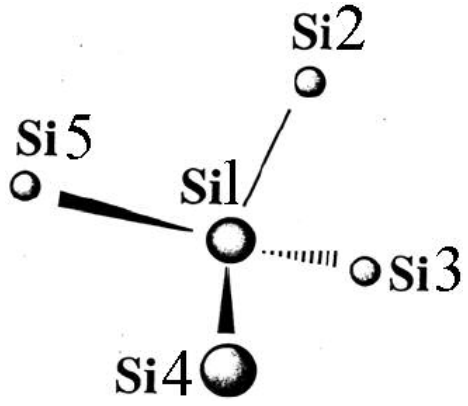


Table 6 lists Cartesian coordinates of 5 atoms.

Table 6

	X	Y	Z
Si_1	0.10595	0.00086	-0.00393
Si_2	-1.49416	-0.39903	1.77593
Si_3	-0.5682	2.10462	-0.89392
Si_4	2.33901	-0.0196	0.66507
Si_5	-0.3826	-1.68685	-1.54315

2. Calculate bond distances of atoms Si_2, Si_3, Si_4, and Si_5 from the central atom Si_1. Sort and order the bond distances and using bond Si_1-Si_2 as the reference bond calculate bond angles θ_{312} (angle between Si_3-Si_1-Si_2), θ_{412}, and θ_{512}. Also calculate dihedral angles between atoms Si_4-Si_1-Si_2-Si_3 (Φ_{4123}) and Si_5-Si_1-Si_2-Si_4 (Φ_{5124}), looking along bond Si_1-Si_2 in Neuman projection.
3. Write a input file for Gaussian-98 specifying the configuration in the Z-matrix format as follows:

```
g98 <<END > si.log
%Mem=200MB
%Chk=si.chk                                          Link0 section
#P B3LYP/6-31G**                                     Route section

Silicon                                              Title section

0   1                                                Molecule
Si1                                                  specification
Si2  Si1  2.426558                                   section
Si3  Si1  2.381677   Si2 103.4742e
Si4  Si1  2.331201   Si2 114.8164e   Si3 123.2472  0
Si5  Si1  2.335859   Si2 103.0822e   Si4 122.7374  0

END
```

First line specifies name for output file,

%Mem command controls the amount of dynamics memory to be used.

Link 0 commands specify scratch file handling and its location.

The *route section* specifies the type of functional and the basis set to be used for the calculation. In the present study B3LYP hybrid functional is used for incorporating electron correlation using 6-31G** basis set.

Molecule specification section specifies the charge, multiplicity and geometry of the molecule. In this example '0' indicates a neutral molecule and '1' indicates multiplicity = 1 (singlet). The geometry is specified in Z-matrix format.

4. Perform *ab initio* calculations for all the configurations stored during the simulation in step 1 to compute the energy and forces using density functional theory with B3LYP procedure to incorporate electron correlation and 6-31G** basis set..
5. Sort and order the input vectors so as to obviate the need to train a network for all possible permutations of a configuration. Scale the input and output vectors so that all the variables fall within a range [-1 +1]. In the present study a neural network with one hidden layer having 9 neurons in the input layer (corresponding to the number of input variables), 45 neurons in the hidden layer, and 1 neuron in the output layer (corresponding to the number of output variables) was used. With 45 neurons in the hidden layer the network has 496 parameters for fitting. The input vectors to the network being internal coordinates and the energies from *ab initio* calculations forming the outputs. To avoid overfitting, use early stopping and Bayesian regularization during training explained in sections 6.12 and 6.13.

6. Use the trained neural network to perform molecular dynamics or Monte Carlo simulation. During the simulation input vectors to the network are ordered and scaled within the range [-1 +1]. The output from the network is post-processed to get the energy/force value. Store more configurations occurring during the simulation. In the present study about 34,000 configurations were stored.
7. Compare the new configurations with the configurations for which the neural network was trained. Modified novelty detection technique based on modified minimum distance computation, compares two input vectors augmented with the output from the network. The purpose of considering the network output is to take into account gradient of the underlying function for sampling procedure.

   Let q_i be an input vector corresponding to the i^{th} configuration, in the current data base. By computing trajectories using the neural network output for the force field a new configuration q_n is generated. Minimum distance computation based on the 2-norm difference between vector q_n and vector q_i is used for the novelty detection.

$$\|q_i - q_n\| = \left[(q_i - q_n)^T \cdot (q_i - q_n)\right]^{1/2}$$

   A vector d is now defined whose elements are the distances between the configurations. The minimum distance separating point q_n from other points in the data base is the minimum element of d denoted as d_m=min(d). The minimum distance between a set of modified configuration vectors is used in place of d_m, to incorporate the effect of the function gradient into the selection algorithm. A modified vector is created as,

$$\Gamma_i = [q_i \; O_i]$$

where O_i is the output from the neural network of the configuration point q_i. When the minimum distance Γ_m between Γ_i and Γ_n is used in place of d_m, the effect of function gradient is incorporated. This is because, if the gradient between points q_i and q_n is large, then the difference between corresponding neural network outputs O_i and O_n will be large, making the magnitude of Γ_m large, and hence the point q_n will be identified as a novelty point. Plot a histogram of minimum distances as shown in Fig 8.6. A large value of minimum distance for outlier configurations would indicate that the outlier configurations are different from the one in the training set and the corresponding network output would be far off from the target values.

Identify the outlier configurations by comparing normalized probability of minimum distances for initial configurations and configurations during iteration as shown in Fig 8.7. Using the selection criterion given in Eqn. (8.3) based on Parzen's distribution select the configurations to be added for retraining the network. In the present study T_1=0.5 and Γ_m=0.06 was used.

8. Perform *ab initio* calculations for the outlier configurations to compute the energy and forces using density functional theory with B3LYP procedure to incorporate electron correlation and 6-31G** basis set.
9. Retrain a neural network for the entire database consisting of initial and outlier configurations as described in step 5.
10. Re-compute the minimum distances of the combined set consisting of initial and outlier configurations to get a histogram plot as shown in Fig 8.8. The re-computed minimum distance values could be less for the concatenated set of

outlier and initial data set of configurations because now the neural network is trained for outlier configurations as well along with initial set of configurations and as a result corresponding output is close to the target values.

11. Use the retrained neural network in the simulation and repeat steps from 5-9 until the minimum distance values are small or lie within the range of minimum distances from the initial set of configurations as shown by the histogram plot of minimum distances as shown in Fig 8.11, which indicates the convergence is obtained over the configuration space involved during the simulation.

VITA

Milind M. Malshe

Candidate for the Degree of

Master of Science

Thesis: *AB INITIO* MOLECULAR DYNAMICS (AIMD)- A NEW APPROACH FOR DEVELOPMENT OF ACCURATE POTENTIALS

Major Field: Mechanical Engineering

Biographical:

Education: Received Bachelor of Engineering in Mechanical Engineering from University of Mumbai, India in 1998. Completed the requirements for the Master of Science degree with a major in Mechanical Engineering at Oklahoma State University in December, 2004.

Experience: Research Assistant at Oklahoma State University from Jan. 2001-present.