A MOLECULAR DYNAMICS SIMULATION STUDY OF TORSIONAL DEFORMATIONS OF SINGLE-WALLED CARBON NANOTUBES

by

Ufuk Paralı

A Thesis Submitted to the
Graduate School of Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computational Sciences and Engineering

Koç University

August, 2005

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Ufuk Paralı

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Dr. Tekin Dereli

_____

Assist. Prof. Dr. Attila Gürsoy

_____

Assist. Prof. Dr. Özgür Müstecaplıoğlu

Date: _____

ABSTRACT

In this study we present the tight binding molecular dynamics (TBMD) simulation results for torsional response of 10x10 single-walled carbon nanotubes for varying numbers of layers, twist angles and temperature conditions. TBMD method we use has O(N) complexity and it has been executed both in sequential and parallel modes. The code is executed in a distributed memory system utilizing the concept of parallel virtual machines (PVM). The mechanical properties of the simulated structure such as bond length distribution, bond angle distribution, torsional deformations, total energy, shear modulus and torsion stiffness were studied under various simulation conditions. We also presented the efficiency and speedup graphs of the parallel computation.

ACKNOWLEDGMENTS

First, I would like to thank my supervisor, Prof. Dr. Tekin Dereli for his guidance, encouragement and support during this research. In addition, I would like to thank Prof. Dr. Gülay Dereli for her insightful suggestions and helpful discussions. I also would like to thank Assist. Prof. Dr. Cem Özdoğan for his help in using the parallel O(N) TBMD simulation code. I would like to thank Banu Süngü and Önder Eyecioğlu for discussions on the computational aspects.

Finally, I would like to express my deepest gratitude to my parents and to my family for their support and encouragement.

TABLE OF CONTENTS

List of Tables **vii**

List of Figures **viii**

Nomenclature **xi**

Chapter 1: Introduction **1**

Chapter 2: Classification of Carbon Nanotubes **5**

2.1 Chiral Vector: C_h 6

2.2 Translational Vector:T 7

2.3 Symmetry Vector: R 8

Chapter 3: Continuum Theory of Stresses and Strains **12**

3.1 Definition of Stress At a Point 12

3.2 Stress Notation 13

3.3 Symmetry Of The Stress Array and Stress On An Arbitrarily Oriented Plane 14

3.4 Differential Equations of Motion of a Deformable Body 16

3.5 Strain Theory 18

3.6 Torsion of Circular Cross-Section Bars 20

3.7 Torsion of a Single Thin-Walled Tube 22

Chapter 4: Computational Techniques for Nanoscale Simulations **24**

4.1 Classical molecular dynamics 25

4.2 Generalized tight-binding molecular dynamics 27

4.3 Ab initio simulation methods 28

4.4 Parallel Computing 30

Chapter 5: Results and Discussion **37**

Chapter 6: Conclusion **44**

Bibliography **45**

Vita **47**

LIST OF TABLES

Table 2.1 : Classification of carbon nanotubes

Table 2.2 : Values for characterization parameters for selected carbon nanotubes labeled by the chiral vector C_h = (n,m)

Table 2.3 : Parameters for carbon nanotubes

LIST OF FIGURES

Figure 2.1 : Classification of carbon nanotubes

Figure 2.2 : The unrolled honeycomb lattice of a nanotube

Figure 2.3 : Space group symmetry operation

Figure 2.4 : Symmetry vector, translation vector, chiral vector reltionship

Figure 3.1 : A general loaded body cut by plane Q

Figure 3.2 : Force transmitted through incremental area of cut body

Figure 3.3 : Body forces

Figure 3.4 : Stress components at a point in a loaded body

Figure 3.5 : General deformed body

Figure 3.6 : Stress components showing changes from face to face along with body force per unit volume including inertial forces

Figure 3.7 : Orthogonal curvilinear coordinates

Figure 3.8 : Line segment PQ in undeformed and deformed body

Figure 3.9 : Line segments PA and PB before and after deformation

Figure 3.10 : Torsion of a circular cross section bar

Figure 3.11 : Observable deformations in torsion

Figure 3.12 : Torsional displacements in a circular bar

Figure 3.13 : Shear stresses in the interior and on the boundary

Figure 3.14 : Stresses in a thin-walled circular cross section

Figure 3.15 : Equilibrium for a single thin walled tube

Figure 3.16 : Equilibrium for relating the torque to the shear stress

Figure 3.17 : Shear stresses and shear flow in a single tube

Figure 5.1 : 10x10, 20 layers, twist angle = 360 deg

Figure 5.2 : 10x10, 20 layers, 300 K

Figure 5.3 : 10x10, 300 K, twist angle = 90 deg

Figure 5.4 : 10x10, 300 K, twist angle = 360 deg

Figure 5.5 : 10x10, 50 layers, 300 K

Figure 5.6 : 10x10, 50 layers, twist angle = 180 deg

Figure 5.7 : 10x10, 20 layers, twist angle = 360 deg, 300 K

Figure 5.8 : 10x10, 20 layers, twist angle = 360 deg, 600 K

Figure 5.9 : 10x10, 20 layers, twist angle = 360 deg, 900 K

Figure 5.10 : 10x10, 20 layers, twist angle = 360 deg, 1200 K

Figure 5.11 : 10x10, 20 layers, twist angle = 45 deg, 300 K

Figure 5.12 : 10x10, 20 layers, twist angle = 90 deg, 300 K

Figure 5.13 : 10x10, 20 layers, twist angle = 120 deg, 300K

Figure 5.14 : 10x10, 40 layers, twist angle = 90 deg, 300 K

Figure 5.15 : 10x10, 60 layers, twist angle = 360 deg, 300 K

Figure 5.16 : 10x10, 50 layers, twist angle = 360 deg, 300 K

Figure 5.17 : 10x10, 50 layers, twist angle = 180 deg, 300 K

Figure 5.18 : 10x10, 50 layers, twist angle = 180 deg, 600 K

Figure 5.19a : 10x10, 20 layers, 300 K, twist angle = 360 deg

Figure 5.19b : 10x10, 20 layers, 300 K, twist angle = 360 deg

Figure 5.20a : 10x10, 20 layers, 600 K, twist angle = 360 deg

Figure 5.20b : 10x10, 20 layers, 600 K, twist angle = 360 deg

Figure 5.21a : 10x10, 20 layers, 900 K, twist angle = 360 deg

Figure 5.21b : 10x10, 20 layers, 900 K, twist angle = 360 deg

Figure 5.22a : 10x10, 20 layers, 1200 K, twist angle = 360 deg

Figure 5.22b : 10x10, 20 layers, 1200 K, twist angle = 360 deg

Figure 5.23a : 10x10, 20 layers, 300 K, twist angle = 45 deg

Figure 5.23b : 10x10, 20 layers, 300 K, twist angle = 45 deg

Figure 5.24a : 10x10, 20 layers, 300 K, twist angle = 90 deg

Figure 5.24b : 10x10, 20 layers, 300 K, twist angle = 90 deg

Figure 5.25 : 10x10, 20 layers, 300 K, twist angle = 120 deg

Figure 5.26a : 10x10, 40 layers, 300 K, twist angle = 90 deg

Figure 5.26b : 10x10, 40 layers, 300 K, twist angle = 90 deg

Figure 5.27a : 10x10, 60 layers, 300 K, twist angle = 360 deg

Figure 5.27b : 10x10, 60 layers, 300 K, twist angle = 360 deg

Figure 5.28 : 10x10, 50 layers, 300 K, twist angle = 360 deg

Figure 5.29 : 10x10, 20 layers, twist angle = 360 deg

Figure 5.30 : 10x10, 20 layers, 300 K

Figure 5.31 : 10x10, 300 K, twist angle = 90 deg

Figure 5.32 : 10x10, 300 K, twist angle = 360 deg

Figure 5.33 : 10x10, 50 layers, 300 K

Figure 5.34 : 10x10, 50 layers, twist angle = 180 deg

Figure 5.35 : Initial Bond-Length Distribution Function For All Cases

Figure 5.36 : Initial Bond-Angle Distribution Function For All Cases

Figure 5.37 : 10x10, 20 layers, twist angle = 360 deg

Figure 5.38 : 10x10, 20 layers, 300 K

Figure 5.39 : 10x10, 300 K, twist angle = 90 deg

Figure 5.40 : 10x10, 300 K, twist angle = 360 deg

Figure 5.41 : 10x10, 50 layers, 300 K

Figure 5.42 : 10x10, 50 layers, twist angle = 360 deg

Figure 5.43 : Shear modulus for 10x10, 20 layers, 300 K CNTs

Figure 5.44 : Shear modulus for 10x10, 20 layers, twist angle = 360 deg CNTs

Figure 5.45 : Torsion stiffness for 10x10, 20 layers, 300 K CNTs

Figure 5.46 : Torsion stiffness for 10x10, 20 layers, twist angle = 360 deg CNTs

Figure 5.47 : Efficiency for 2, 4, 6, 8 processors

Figure 5.48 : Speedup for 2, 4, 6, 8 processors

NOMENCLATURE

SWCN	Single-Walled Carbon Nanotube
MWCN	Multi-Walled Carbon Nanotube
MD	Molecular Dynamics
TBMD	Tight Binding Molecular Dynamics
CNT	Carbon Nanotube
DFT	Density Functional Theory
LDA	Local Density Approximation
PW	Plane Wave
MPPs	Massively Parallel Processors
PVM	Parallel Virtual Machine
SPMD	Single Program Multiple Data
TID	Task Identifier
MPI	Message Passing Interface

Chapter 1

INTRODUCTION

Carbon nanotubes have been the focus of considerable research since their discovery in 1991. They combine exceptional mechanical, thermal and electrical properties. They have high modulus, stronger than steel, thermally stable up to 2800^{o}C in vacuum, thermal conductivity is twice that of diamond, and the electric - current carrying capacity is 1000 times higher than that of copper wire. All of these properties open up the broad application areas for carbon nanotubes [1]. In the simplest terms, carbon nanotubes can be thought of as rolled up, closed graphite sheets. This sheet can be rolled up at different discrete angles to create SWNTs. Thus, an SWNT is a single, closed molecule with few to no atomic imperfections, and a hexagonal ring bonding structure similar to graphite. The ring bonding structure, specifically the hybridized sigma bonds (sp2) imparts the impressive mechanical properties. Multi-walled nanotubes also exist, consisting of nested SWNTs 3.4 Angstrom apart, which widen the range of tube properties and application possibilities. The single-wall nanotubes are interesting nanoscale materials for the following four reasons:

1) Single-wall (and also multi-wall) nanotubes have very good elasto-mechanical properties due to the two-dimensional arrangement of carbon atoms in a graphene sheet that allows large out-of-plane distortions, while the strength of carbon-carbon in-plane bonds keeps the graphene sheet exceptionally strong against any in-plane distortion or fracture. These structural and material characteristics of nanotubes point towards their possible use in making next generation of extremely lightweight, but highly elastic, and very strong composite materials.

2) A single-wall nanotube can be either conducting or semiconducting, depending on its chiral vector (n, m), where n and m are two integers. All armchair CNTs are conducting and zig-zag CNTs are semiconducting. The rule is that when the difference n-m is a multiple of three, a conducting nanotube is obtained. If the difference is not a multiple of three, a

semiconducting nanotube is obtained. In addition, it is also possible to connect nanotubes with different chiralities creating nanotube hetero-junctions, which can form a variety of nanoscale molecular electronic device components.

3) Nanotubes, by structure, are high aspect-ratio objects with good electronic and mechanical properties. Consequently, the applications of nanotubes in field-emission displays or scanning probe microscopic tips for metrological purposes, have started to materialize even in the commercial sector.

4) Since nanotubes are hollow, tubular, caged molecules, they have been proposed as lightweight large surface area packing material for gas-storage and hydrocarbon fuel storage devices, and gas or liquid filtration devices, as well as nanoscale containers for molecular drug-delivery and casting structures for making nanowires and nanocapsulates [4], [5].

Due to the high specific stiffness and strength, nanotubes represent a very promising material as reinforcements in composite materials. Elastic properties of both multi- and single-walled nanotubes (SWNT) have been investigated extensively through experimentation and theoretical approaches. Experimental studies have been performed to characterize nanotube materials. Due to the difficulty in performing these experiments, the scatter in the results is considerable. The shear stiffness of carbon nanotubes was found to lie between 0.5 - 4 TPa while the tensile modulus and strength of nanotubes have been reported to range from 0.27 TPa to 3.6 TPa and 11 to 200 GPa, respectively. Stating these values underscores two common continuum assumptions. First that they are linear elastic materials and second they have a thickness corresponding to density of graphite (thickness = 0.34 nm). Naturally, with such poor resolution, the validity of these assumptions can not be checked with experiment [4], [8]. Theoretcal studies in this respect are essential.

Theoretical approaches can be classified into two categories: namely the "bottom up" approach based on quantum/molecular mechanics including the classical molecular dynamics (MD) and ab initio methods, and the "top down" approach based on continuum mechanics. In general, ab initio methods give more accurate results than MD, but they are also much more computationally expensive (only suitable for small systems containing at most hundreds of atoms). Despite constant increases in available computational power and improvement in numerical algorithms, even classical molecular dynamics computations are still limited to simulating on the order of $10^6 - 10^8$ atoms for a few nanoseconds. The

simulation of larger systems or longer times must currently be left to continuum methods. However, at the nanoscale, theories for describing continuum materials have reached their limit. The accuracy of using these continuum theories becomes questionable in many of the most interesting cases of nanomechanics

Atomic simulations have proven to be a good vehicle for studying nanotube mechanical responses. Molecular mechanics has been used to obtain Young's moduli of 0.75 - 1.25 TPa, using the same assumptions as described above. Molecular mechanics is a method of modeling the interatomic forces, including bonding and non-bonded forces, with simple polynomial and trigonometric expansions. This method was developed for the modeling of large molecules such as proteins and lends itself well to the study of nanotubes. The advantage of molecular mechanics is the precision with which tests can be performed. Not only can virtual tension tests be performed, but other tests such as torsion can be performed which are not yet feasible experimentally. The accuracy of these methods vary, but have been shown to predict moduli with the same precision as quantum mechanical methods [4].

In this study, a tight-binding molecular dynamics (TBMD) method has been used to understand the torsional deformations of 10x10 single-walled carbon nanotubes in different twist angle and temperature conditions with different layer values. The tight binding theory of electronic structure has played an increasingly important role in computational materials science. It has developed as an effective tool for calculations of atomic and electronic structures, total energies, diffusion barriers and inter atomic forces of large condensed matter and molecular systems. The advantages of TB theory include the ease of implementation, low computational workload, robust transferability as well as relatively good reliability. Traditional TB solves the Schrdinger equation in reciprocal space by direct matrix diagonalization, which results in cubic scaling with respect to the number of atoms. The complexity of the applied TBMD method is O(N). This O(N) algorithm has been executed both in sequential and in parallel modes which is the main aim of this study. The O(N) methods solve for the band energy in real space and make the approximation that only the local environment contributes to the bonding, and hence band energy, of each atom. It is worth noting that usually O(N) schemes can be efficiently parallelized through the use of message passing libraries [7], [9].

At the end of this computational study the following results have been obtained for the

investigated structure for different temperature, twist angle and layer values : Total energy, bond-length distribution, bond-angle distribution, torsional deformations, shear modulus, torsion stiffness, efficiency and speedup values. The necessary comparisons have been done and commented in this study.

In Chapter 2, geometrical classification of carbon nanotubes is given. In Chapter 3, continuum elastic theory of stresses and strains and the mechanics of torsion are presented. In Chapter 4, computational techniques for nanoscale computations are reviewed. In Chapter 5, the physical results of nanoscale computations are discussed. We conclude in Chapter: 6.

Chapter 2

CLASSIFICATION OF CARBON NANOTUBES

A single-walled carbon nanotube can be described as a graphene sheet rolled into a cylindrical shape so that the structure is one-dimensional with axial symmetry, and in general exhibiting a spiral conformation, called chirality. The chirality, as defined in this chapter, is given by a single vector called the chiral vector. To specify the structure of carbon nanotubes, several important vectors are defined, which are derived from the chiral vector. The references for the following review are [2] and [3].

A single-walled nanotube is defined by a cylindirical graphene sheet with a diameter of about 0.7 - 10.0 nm, though most of the observed single-walled nanotubes have diameters < 2 nm. If we neglect the two ends of a carbon nanotube and focus on the large aspect ratio of the cylinder (i.e., length/diameter which can be as large as $10^4 - 10^5$), these nanotubes can be considered as one-dimensional nanostructures. An interesting and essential fact about the structure of a carbon nanotube is the orientation of the six-membered carbon ring (hereafter called a hexagon) in the honeycomb lattice relative to the axis of the nanotube. Three examples of single-walled carbon nanotubes (SWCN's) are shown in Figure 2.1. From this figure, it can be seen that the direction of the six-membered ring in the honeycomb lattice can be taken almost arbitrarily, without any distortion of the hexagons except for the distortion due to the curvature of the carbon nanotube. This fact provides many possible structures for carbon nanotubes, even though the basic shape of the carbon nanotube wall is a cylinder. In Figure 2.1 we show the terminations of each of the three nanotubes. The terminations are often called caps and consist of a "hemisphere" of a fullerene. Each cap contains six pentagons and an appropriate number and placement of hexagons that are selected to fit perfectly to the long cylindrical section. In this chapter we focus on the periodic structure along the nanotube axis. The primary symmetry classification of a carbon nanotube is as either being chiral or achiral. An achiral carbon nanotube is defined by a carbon nanotube whose mirror image has an identical structure to the original one.

There are only two cases of achiral nanotubes; armchair and zigzag nanotubes, as are shown in Figure 2.1 (a) and (b), respectively. The names of armchair and zigzag arise from the shape of the cross-sectional ring, as is shown at the edge of the nanotubes in Figure 2.1 (a) and (b) respectively. Chiral nanotubes exhibit a spiral symmetry whose mirror image cannot be superposed on to the original one. We call this tube a chiral nanotube, since such structures are called axially chiral in the chemical nomenclature. Axial chirality is commonly discussed in connection with optical activity. We have thus a variety of geometries in carbon nanotubes which can change with diameter, chirality and cap structures. A classification of carbon nanotubes is given in Table 2.1.

2.1 Chiral Vector: C_h

The structure of a single-walled carbon nanotube is specified by the vector (OA in figure 2.2) which corresponds to a section of the nanotube perpendicular to the nanotube axis (the equator of the nanotube). In Figure 2.2, the unrolled honeycomb lattice of the nanotube is shown, in which OB is the direction of the nanotube axis, and the direction of OA corresponds to the equator. By considering the crystallographically equivalent sites O, A, B, and B', and by rolling the honeycomb sheet so that points O and A coincide (and points B and B' coincide), a paper model of carbon nanotube can be constructed. The vectors OA and OB define the chiral vector C_h and the translational vector T of a carbon nanotube, respectively. The chiral vector C_h can be expressed by the real space unit vectors a_1 and a_2 (see Figure 2.2) of the hexagonal lattice:

$$C_h = na_1 + ma_2 \equiv (n, m), (n, m \; are \; integers, \; 0 \leq |m| \leq n). \tag{2.1}$$

The specific chiral vectors C_h for the carbon nanotubes shown in Figure 2.1 are, respectively, (a) (5,5), (b) (9,0), and (c) (10,5), and the chiral vector shown in Figure 2.2 is (4.2). As is shown in Table 2.1, an armchair nanotube corresponds to the case of n=m, that is C_h=(n,n), and a zigzag nanotube corresponds to the case of m=0, or C_h=(n,0). All other (n,m) chiral vectors correspond to chiral nanotubes. Because of the hexagonal symmetry of the honeycomb lattice, we need to consider only $0 < |m| < n$ in C_h=(n,m) for chiral nanotubes.

The diameter of the carbon nanotube is

$$d_t = L/\pi, L = |C_h| = \sqrt{C_h.C_h} = a\sqrt{n^2 + m^2 + nm} \tag{2.2}$$

where L is the circumferential length of the carbon nanotube. It is noted here that a_1 and a_2 need not be orthogonal to each other and that the inner products between a_1 and a_2 yield:

$$a_1.a_1 = a_2.a_2 = a^2,\ a_1.a_2 = \frac{a^2}{2} \tag{2.3}$$

where the lattice constant $a = 1.44A \times \sqrt{3} = 2.49A$ for the honeycomb lattice. The C-C bond length of graphite is $1.42A$. In the case of carbon nanotubes, the C-C bond length is known to be slightly larger than graphite: $1.44A$ [3].

The chiral angle θ (Figure 2.2) is defined as the angle between the vectors C_h and a_1, with values of θ in the range $0 \leq |\theta| \leq 30$, because of the hexagonal symmetry of the honeycomb lattice. The chiral angle θ denotes the tilt angle of the hexagons with respect to the direction of the nanotube axis, and the angle θ specifies the spiral symmetry. The chiral angle θ is defined by taking the inner product of C_h and a_1, to yield an expression for cos θ:

$$cos\ \theta = \frac{C_h.a_1}{|C_h||a_1|} \tag{2.4}$$

thus relating θ to the integers (n,m) defined in equation 2.1. In particular, zigzag and armchair nanotubes correspond to θ=0 and θ=30, respectively [3].

2.2 Translational Vector:T

The translational vector T is defined to be the unit vector of a 1D carbon nanotube. The vector T is parallel to the nanotube axis and is normal to the chiral vector C_h in the unrolled honeycomb lattice in Figure:2.2. The lattice vector T shown as OB in Figure:2.2 can be expressed in terms of the basis vectors a_1 and a_2 as:

$$T = t_1a_1 + t_2a_2 \equiv (t_1, t_2). \tag{2.5}$$

The translation vector T corresponds to the first lattice point of the 2D graphene sheet through which the vector OB (normal to the chiral vector C_h) passes. From this fact, it is clear that t_1 and t_2 do not have a common divisor except for unity. Using C_h.T=0 and

equations 2.1, 2.3, and 2.5, we obtain expressions for t_1 and C_h given by:

$$t_1 = \frac{2m+n}{d_R},\ t_2 = -\frac{2n+m}{d_R} \tag{2.6}$$

where d_R is the greatest common divisor (gcd) of (2m+n) and (2n+m). Furthermore, by introducing d as the greatest common divisor of n and m, then d_R can be related to d:

$$d_R = \begin{cases} d & \text{if} \quad (n-m) \neq \text{multiple of } 3d \\ 3d & \text{if} \quad (n-m) = \text{multiple of } 3d \quad . \end{cases} \tag{2.7}$$

In the case of Figure:2.2, where C_h=(4,2), we have d=d_R=2, T=(4,-5). The length of the translation vector, T, is given by:

$$|T| = \sqrt{3}L/d_R \tag{2.8}$$

where the circumferential nanotube length L is given by equation 2.2. We note that the length T is greatly reduced when (n,m) have a common divisor or when (n-m) is a multiple of 3d. In fact, for the C_h=(5,5) armchair nanotube, we have d_R=3d=15, T=(1,-1) (Figure:2.1a), while for the C_h=(9,0) zigzag nanotube we have d_R=d=9, and T=(1,-2) (Figure:2.1b) [3]

The unit cell of the 1D carbon nanotube is the rectangle $OAB'B$ defined by the vectors C_h and T (Figure:2.2), while the vectors a_1 and a_2 define the area of the unit cell of 2D graphite. When the area of the nanotube unit cell $|C_h \times T|$ (where the symbol $\times$ denotes the vector product operator) is divided by the area of a hexagon $|a_1 \times a_2|$, the number of hexagons per unit cell N is obtained as a function of n and m as in equation 2.1 as:

$$N = \frac{|C_h.T|}{|a_1 x a_{2|}} = \frac{2(m^2+n^2+nm)}{d_R} = \frac{2L^2}{a^2 d_R} \tag{2.9}$$

where L and d_R are given by equations 2.2 and 2.7, respectively, and we note that each hexagon contains two carbon atoms. Thus there are 2N carbon atoms in each unit cell of the carbon nanotube [3].

2.3 Symmetry Vector: R

The carbon atom site vectors are denoted within the 1D nanotube unit cell by i times the vector R, that is, iR, where i is an integer (i=1...N). When iR goes out of the unit cell, we shift it to lie within the unit cell through translation by an integral number of C_h or

T vectors, using periodic boundary conditions. The vector R is used for generating the coordinates of carbon atoms in the nanotube. It is convenient to express the R vector in terms of its projections on the orthogonal vectors C_h and T of the nanotube unit cell, as shown in Figure 2.3. The symmetry vector R is then defined as the site vector (shown by OR in Figure 2.2) having the smallest component in the direction of C_h, and R is expressed in terms of a_1 and a_2 as:

$$R = pa_1 + qa_2 \equiv (p, q) \tag{2.10}$$

where p and q do not have a common divisor except for unity and they are integers. The C_h component of R, or C_h.R, is proportional to the value of T $\times$R given by:

$$T \times R = (t_1 q - t_2 p)(a_1 x a_2) \tag{2.11}$$

where $(t_1 q - t_2 p)$ on the right hand side is an integer. We select p and q of R to form the smallest site vector (i=1), such that

$$t_1 q - t_2 p = 1, \ (0 < mp - nq \leq N) \tag{2.12}$$

The solution of equation 2.12 for p and q is uniquely determined if t_1 and t_2 of equation 2.5 do not have a common divisor except for unity. The second condition in equation 2.12, $0 < mp - nq < N$, arises from the fact that R exists within the 1D nanotube unit cell, so that

$$0 < \frac{R.T}{T^2} = \frac{|C_h x R|}{LT} = \frac{mp - nq}{N} < 1 \tag{2.13}$$

using equations 2.2, 2.8, and 2.9. Similarly, using equations 2.6 and 2.9, we obtain another necessary condition arising from R being within the 1D unit cell:

$$0 < \frac{R.C_h}{L^2} = \frac{|RxT|}{LT} = \frac{t_1 q - t_2 p}{N} \leq 1 \tag{2.14}$$

and from equation 2.14, we get the condition,

$$0 < t_1 q - t_2 p \leq N. \tag{2.15}$$

Since the first condition of equation 2.12 satisfies equation 2.14, it is not necessary to add this condition to the definition of R [3].

To determine all N site location vectors iR, (i=1...N) of the nanotube unit cell, we use expression i(t_1q-t_2p)=i for each i, and note that the maximum value of i(t_1q-t_2p) becomes

N. Using the fact that the C_h component of NR is always equal to $|C_h|$=L, the vectors iR define N inequivalent sites in the nanotube unit cell, and will thus have different values for their projections along the direction of C_h. Therefore iR, (i=1...N), uniquely generates N different atom sites in the unit cell of the nanotube.

Taking the indicated vector products $R \times C_h$ and $R \times T$ and using equations 2.2, 2.8, 2.9, and 2.12, we obtain the expressions for the length of τ and the rotation angle φ as :

$$\tau = \frac{|R \times C_h|}{L} = \frac{(mp - nq)|a_1 \times a_2|}{L} = \frac{(mp - nq)T}{N},$$
$$\varphi = \frac{|T \times R|}{T}\frac{2\pi}{L} = \frac{d_R(t_1 q - t_2 p)}{\sqrt{3}L}\frac{\sqrt{3}a^2}{2} = \frac{2\pi}{N}. \tag{2.16}$$

Using these definitions, the rotation angle φ becomes 2π/N, where N is the number of hexagons in the 1D unit cell of the nanotube given by equation 2.9 [3].

According to Figure 2.3, the symmetry operation that brings the lattice point O to an equivalent lattice point C is shown in Figure 2.4. Here

$$NR = C_h + MT \tag{2.17}$$

and where

$$M \equiv mp - nq \tag{2.18}$$

is an integer which denotes the number of T vectors that are necessary for reaching the distance from O to NR [3].

In Table 2.2 the characteristic parameters of carbon nanotubes are listed specified by (n,m), including d, the greatest common divisor of n and m, and the related quantity d_R which is given by equation 2.7. Also listed in table 2.2 are the nanotube diameters d_t in units of A, the lengths of the chiral vector L and of the translation repeat distance T of the 1D lattice (both in units of the lattice constant $a = \sqrt{3}a_{C-C}$ for 2D graphene sheet), the number N of hexagons per unit cell of the 1D nanotube, the translation vector T, the symmetry vector R and the integer M. The basic symmetry operation $R = (\varphi|\tau)$ can be obtained from Table 2.2, equation 2.16 and values for $T = (t_1, t_2)$ and $R = (p, q)$.

To show the use of Table 2.2, take the $C_h = (4, 2)$ nanotube shown in Figure 2.2, which has $T = (4, -5), R = (1, -1), N = 28, d = d_R = 2, T = \sqrt{21}a, L = \sqrt{28}a, \tau = 6T/28$ and $M = 6$ translations of vector T to reach the point C in Figure 2.4. For the case

of the nanotube $C_h = (7, 4)$ which is a chiral nanotube with $n - m = 3$, there are no common divisors, so $d = 1$, but since $n - m = 3$, we have $d_R = 3$. Thus we obtain $L = \sqrt{93}a, T = \sqrt{31}a, N = 62$ and $M = 11$ translations of the vector T. For the armchair nanotube (5,5), which needs half C_{60} fullerenes to form its end caps, the highest common divisor is 5, and since n-m=0, we have $d_R = 3{\times}5 = 15$, yielding $N = 10, \varphi = 2\pi/10, \tau = T/2$, and $M = 9$ [3].

All parameters defined in this section are summarized in Table 2.3. The values of all parameters listed here depend on the two integers, n and m, of the chiral vectors C_h.

Chapter 3

CONTINUUM THEORY OF STRESSES AND STRAINS

Since in this study, torsion response of CNTs is studied, we must understand the concept of strain energy stored in the system. The source of the stored strain energy is the deformation of the system due to torsion. In order to understand the strain energy, we must understand the concept of theory of stresses and strains. The torsion of SWNT is modeled as the torsion of a thin walled tube by using continuum elasticity theory and the calculations of shear modulus and torsion stiffnes is calculated with respect to this model. The references for this review are [10], [11], [12].

3.1 Definition of Stress At a Point

Consider a general body subjected to forces acting on its surface Figure 3.1. Pass a fictitious plane Q through the body, cutting the body along surface A (Figure 3.2). Assume that one side of plane Q as positive and the other side as negative. The portion of the body on the positive side of Q exerts a force on the portion of the body on the negative side. This force is transmitted through the plane Q by direct contact of the parts of the body on the two sides of Q. The force is transmitted through an incremental area of ΔA of A by the part on the positive side Q, denoted by ΔF. According to the Newton's third law, the portion of the body on the negative side of Q transmits through area ΔA a force -ΔF.

The force ΔF may be resolved into components ΔF_N and ΔF_S, along the unit normal N and unit tangent S, respectively, to the plane Q. The force ΔF_N is called the normal force on area ΔA and ΔF_S is called the shear force on ΔA. The forces ΔF, ΔF_N, and ΔF_S depend on the area ΔA and the orientation of plane Q. The magnitudes of the average forces per unit area ΔF/ΔA, ΔFN/ΔA, and ΔFS/ΔA approach limits different from zero in general. The limiting ratio of $\Delta F/\Delta A$ as ΔA goes to zero defines the stress vector σ. Thus, the stress vector σ is given by

$$\sigma = \lim_{\Delta A \to 0} \frac{\Delta F}{\Delta A} \tag{3.1}$$

The stress vector σ always lies along the limiting direction of force vector ΔF, which in general is neither perpendicular nor tangent to the plane Q.

Similarly the limiting ratios of $\Delta F_N/\Delta A$ and $\Delta F_S/\Delta A$ define the normal stress vector σ_{N} and the shear stress vector σ_{S} that act at a point in the plane Q. These stress vectors are defined by the relations

$$\sigma_N = \lim_{\Delta A \to 0} \frac{\Delta F_N}{\Delta A}, \tag{3.2}$$

$$\sigma_S = \lim_{\Delta A \to 0} \frac{\Delta F_S}{\Delta A}. \tag{3.3}$$

The unit vectors associated with σ_N and σ_S are perpendicular and tangent, respectively, to the plane Q.

3.2 Stress Notation

Free body diagrams are used to specify the state of stress at a point and to obtain relations between various stress components. In general a free body diagram may be a diagram of a complete member, a portion of the member obtained by passing a cutting plane through the member, or a boxlike volume element of the member. The loads that act on any of these free bodies can be divided into two types as follows:

1) Surface forces, which are forces that act on the surface of the free body. 2) Body forces, which are forces that act throughout the volume of that portion of the member considered in the free body diagram.

Examples of surface forces are contact forces and distributed loads. Concentrated loads and reaction at a point are considered contact forces. Distributed loads may be either line loads with dimensions of force per unit length or surface loads with dimensions of force per unit area (dimensions of pressure or stress). Distributed loads on beams are often indicated as loads per unit length. Examples of surface loads are pressure exerted by a fluid in contact with the body or shear stresses that act on a cut section of the body. [10], [11], [12]

Examples of body forces are gravitational forces, magnetic forces and inertia forces. Since the body force is distributed through out the volume of the free body, it is convenient to define body force per unit volume. For body force per unit volume, the notation B or B_x, B_y, B_z, where B stands for body and subscripts (x,y,z) denote components in the (x,y,z) directions, respectively, of the rectangular coordinate system (x,y,z) (Figure:3.3).

In Figure:3.4, a free body diagram of a box shaped volume element at a point O in a member is shown, with sides parallel to the (x,y,z) axes. It is assumed that the stress components are uniform (constant) throughout the volume element. The surface forces are represented by the product of the stress components and the areas on which they act. Body forces, represented by the product of the components (B_x, B_y, B_z) and the volume of the element (product of the three infinitesimal lengths of the sides of the element), are higher order terms and are not shown on the free body diagram in Figure 3.4. The face from which the positive x axis is extended is taken to be the positive face; the opposite face perpendicular to the x axis is taken to be the negative face. The stress components σ_{xx}, σ_{xy} and σ_{xz} acting on the positive face are taken to be in the positive sense as shown when they are directed in the positive x, y and z directions.

According to Newton's third law, the positive stress components σ_{xx}, σ_{xy} and σ_{xz} shown acting on the negative face in Figure 3.4 are in the negative (x,y,z) directions, respectively. A positive stress component σ_{xx} exerts a tension (pull) parallel to the x axis. There are nine components of stress exist at point O:

$(\sigma_{xx}, \sigma_{xy}, \sigma_{xz})$, $(\sigma_{yy}, \sigma_{yx}, \sigma_{yz})$, $(\sigma_{zz}, \sigma_{zx}, \sigma_{zy})$.

3.3 Symmetry Of The Stress Array and Stress On An Arbitrarily Oriented Plane

The nine stresses with respect to a rectangular coordinate axes (x,y,z) can be shown in array form as follows:

$$t = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{pmatrix} \tag{3.4}$$

where t symbolically represents the stress array called the stress tensor. In this array, the stress components in the first, second, and third rows act on planes perpendicular to the (x,y,z) axes, respectively. These nine stress components are required to describe the state of stress at a point in a member. However, if the only forces that act on the free body in Figure 3.4 are surface forces and body forces, from the equilibrium of volume element in Figure 3.4, the three pairs of the shear stresses are equal. Summation of moments gives the

following results:

$$\sigma_{yz} = \sigma_{zy} \tag{3.5}$$

$$\sigma_{zx} = \sigma_{xz} \tag{3.6}$$

$$\sigma_{xy} = \sigma_{yx} \tag{3.7}$$

So, according to above results, we can write (3.4) as below:

$$\begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{pmatrix} \tag{3.8}$$

For this type of stress theory, only six components of stress are required to describe the state of stress at a point in a member.

The stress vectors σ_x, σ_y and σ_z on planes that are perpendicular, respectively, to the the x, y and z axes are

$$\sigma_x = \sigma_{xx}i + \sigma_{xy}j + \sigma_{xz}k \tag{3.9}$$

$$\sigma_y = \sigma_{yx}i + \sigma_{yy}j + \sigma_{yz}k \tag{3.10}$$

$$\sigma_z = \sigma_{zx}i + \sigma_{zy}j + \sigma_{zz}k \tag{3.11}$$

where i, j and k are unit vectors relative to the (x,y,z) axes (σ_x on Figure 3.5). Consider the stress vector σ_P on an arbitrary oblique plane P through point O of a member (Figure 3.6). The unit normal vector to plane P is

$$N = li + mj + nk \tag{3.12}$$

where (l,m,n) are direction cosines of the unit vector N. Vectorial summation of forces acting on the tetrahedral element of OABC gives

$$\sigma_p = l\sigma_x + m\sigma_y + n\sigma_z. \tag{3.13}$$

In terms of projections (σ_{Px}, σ_{Py}, σ_{Pz}) of the stress vector σ_P along the axes (x,y,z), it can be written as

$$\sigma_p = \sigma_{Px}i + \sigma_{Py}j + \sigma_{Pz}k \tag{3.14}$$

Comparison of (3.13) and (3.14) gives:

$$\sigma_{Px} = l\sigma_{xx} + m\sigma_{yx} + n\sigma_{zx}, \tag{3.15}$$

$$\sigma_{Py} = l\sigma_{xy} + m\sigma_{yy} + n\sigma_{zy}, \tag{3.16}$$

$$\sigma_{Pz} = l\sigma_{xz} + m\sigma_{yz} + n\sigma_{zz}. \tag{3.17}$$

Equations (3.15), (3.16), (3.17) allow the computation of components of stress on any oblique plane defined by the unit normal N:(l,m,n), provided that the six components of stress

$$\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy} = \sigma_{yx}, \sigma_{xz} = \sigma_{zx}, \sigma_{yz} = \sigma_{zy} \tag{3.18}$$

at point O are known. When point O lies on the surface of the member where the surface forces are represented by distribution of normal and shear stresses, equations (3.15), (3.16), (3.17) represent the stress boundary conditions at point O.

3.4 Differential Equations of Motion of a Deformable Body

These equations are needed when the theory of elasticity is used to derive load-stress and load-deflection relations for a member. In Figure 3.5, a general deformed body is considered and a differential volume element at point O is chosen. The form of the differential equations of motion depends on the type of orthogonal coordinate axes employed. Rectangular coordinate axes (x,y,z) whose directions are parallel to the edges of the volume element is chosen. Since our concentration is mainly on small displacements, we do not distinguish between coordinate axes in the deformed state and in the undeformed state. Six cutting planes bound the volume element shown as a free-body diagram in Figure: 3.6. In general, the state of stress changes with the location of point 0.In particular, the stress components undergo changes from one face of the volume element to another face. Body forces (B_x, B_y, B_z) are included in the free body diagram.

To write the differential equations of motion, each stress component must be multiplied by the area on which it acts and each body force must be multiplied by the volume element since (B_x, B_y, B_z) have dimensions of force per unit volume. The equations of motion for the volume element in Figure 3.6 are then obtained by summation of these forces and summation of moments. In Equations (3.5), (3.6), (3.7) summation of moments has already been used to obtain the stress symmetry conditions. Summation of forces in the x direction gives :

$$\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{yx}}{\partial y} + \frac{\partial \sigma_{zx}}{\partial z} + B_x = 0 \tag{3.19}$$

where σ_{xx}, σ_{yx}=σ_{xy}, and σ_{xz}=σ_{zx} are stress components in the x direction and B_x is the body force per unit volume in the x direction including inertial (acceleration) forces. Summation of forces in the y and z directions give similar results. The three equations of motions are:

$$\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{yx}}{\partial y} + \frac{\partial \sigma_{zx}}{\partial z} + B_x = 0, \tag{3.20}$$

$$\frac{\partial \sigma_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{zy}}{\partial z} + B_y = 0, \tag{3.21}$$

$$\frac{\partial \sigma_{xz}}{\partial x} + \frac{\partial \sigma_{yz}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} + B_z = 0. \tag{3.22}$$

The form of differential equations of motion depends on the coordinate axes. Equations (3.20), (3.21), (3.22) are derived for rectangular coordinate axes. Below is the most general form of the differential equations of motion relative to orthogonal curvilinear coordinates (x, y, z):

$$\frac{\partial(\beta\gamma\sigma_{xx})}{\partial x} + \frac{\partial(\gamma\alpha\sigma_{yx})}{\partial y} + \frac{\partial(\alpha\beta\sigma_{zx})}{\partial z} + \gamma\sigma_{yx}\frac{\partial \alpha}{\partial y} + \beta\sigma_{zx}\frac{\partial \alpha}{\partial z} - \gamma\sigma_{yy}\frac{\partial \beta}{\partial x} - \beta\sigma_{zz}\frac{\partial \gamma}{\partial x} + \alpha\beta\gamma B_x = 0 \tag{3.23}$$

$$\frac{\partial(\beta\gamma\sigma_{xy})}{\partial x} + \frac{\partial(\gamma\alpha\sigma_{yy})}{\partial y} + \frac{\partial(\alpha\beta\sigma_{zy})}{\partial z} + \alpha\sigma_{zy}\frac{\partial \beta}{\partial z} + \gamma\sigma_{xy}\frac{\partial \beta}{\partial x} - \alpha\sigma_{zz}\frac{\partial \gamma}{\partial y} - \gamma\sigma_{xx}\frac{\partial \alpha}{\partial y} + \alpha\beta\gamma B_y = 0 \tag{3.24}$$

$$\frac{\partial(\beta\gamma\sigma_{xz})}{\partial x} + \frac{\partial(\gamma\alpha\sigma_{yz})}{\partial y} + \frac{\partial(\alpha\beta\sigma_{zz})}{\partial z} + \beta\sigma_{xz}\frac{\partial \gamma}{\partial x} + \alpha\sigma_{yz}\frac{\partial \gamma}{\partial y} - \beta\sigma_{xx}\frac{\partial \alpha}{\partial z} - \alpha\sigma_{yy}\frac{\partial \beta}{\partial z} + \alpha\beta\gamma B_z = 0 \tag{3.25}$$

where (α, β, γ) are metric coefficients that are functions of the coordinates (x, y, z). They are defined by:

$$ds^2 = \alpha^2 dx^2 + \beta^2 dy^2 + \gamma^2 dz^2 \tag{3.26}$$

where ds is the differential arc length representing the diagonal of a volume element (Figure 3.7) with edge lengths $\alpha dx, \beta dy$, and γdz, and where (B_x, B_y, B_z) are the components of body force per unit volume including inertial forces. For rectangular coordinates, α=β=γ=1 and equations (3.23), (3.24), (3.25) reduce to equations (3.20), (3.21), (3.22).

To obtain a cylindrical coordinate system, let $x = r, y = \theta, z = z$. The differential length ds is defined by the relation

$$ds^2 = dr^2 + r^2 d\theta^2 + dz^2 \tag{3.27}$$

From which we obtain

$$\alpha = 1, \beta = r, \gamma = 1 \tag{3.28}$$

Substituting (3.28) into equations (3.23), (3.24), (3.25), the following differential equations of motion are obtained:

$$\frac{\partial \sigma_{rr}}{\partial r} + \frac{1}{r}\frac{\partial \sigma_{\theta r}}{\partial \theta} + \frac{\partial \sigma_{zr}}{\partial z} + \frac{\sigma_{rr} - \sigma_{\theta\theta}}{r} + B_r = 0 \tag{3.29}$$

$$\frac{\partial \sigma_{r\theta}}{\partial r} + \frac{1}{r}\frac{\partial \sigma_{\theta\theta}}{\partial \theta} + \frac{\partial \sigma_{z\theta}}{\partial z} + \frac{2\sigma_{r\theta}}{r} + B_\theta = 0 \tag{3.30}$$

$$\frac{\partial \sigma_{rz}}{\partial r} + \frac{1}{r}\frac{\partial \sigma_{\theta z}}{\partial \theta} + \frac{\partial \sigma_{zz}}{\partial z} + \frac{\sigma_{rz}}{r} + B_z = 0 \tag{3.31}$$

where (σ_{rr}, $\sigma_{\theta\theta}$, σ_{zz}, $\sigma_{r\theta}$, σ_{rz}, $\sigma_{\theta z}$) represent stress components defined relative to cylindrical coordinates (r, θ, z).

3.5 Strain Theory

The theory of stresses of a continuous medium depends only on Newton's laws. The theory of strains depends only on geometrical concepts. Both the theories of stresses and strains are independent of material behavior, they are applicable to the study of all materials. Although the theories of stresses and strains are based on different physical concepts, mathematically, they are equivalent.

When a body is deformed, the particle at point P(x,y,z) goes to the point $P^*(x^*, y^*, z^*)$(Figure 3.8). Also the particle at point $Q(x+dx, y+dy, z+dz)$ goes to the point $Q^*(x^*+dx^*, y^*+dy^*, z^*+dz^*)$, and the infinitesimal line element $PQ = ds$ passes into the line element $P^*Q^* = ds^*$.The engineering strain ϵ_E of the line element $PQ = ds$ is given by

$$\epsilon = \frac{ds^* - ds}{ds}. \tag{3.32}$$

Therefore, by definition, $\epsilon_E > -1$. The total differential for dx^* is obtained as:

$$dx^* = \frac{\partial x^*}{\partial x}dx + \frac{\partial x^*}{\partial y}dy + \frac{\partial x^*}{\partial z}dz \tag{3.33}$$

with similar expressions for dy^* and dz^*. Here,

$$x^* = x + u \tag{3.34}$$

$$y^* = y + v \tag{3.35}$$

$$z^* = z + w \tag{3.36}$$

where (u,v,w) are the (x,y,z) components of the displacement of P to P^*. Also

$$ds^2 = dx^*2 + dy^2 + dz^2 \tag{3.37}$$

$$ds^{*2} = dx^{*2} + dy^{*2} + dz^{*2} \tag{3.38}$$

So, we can determine the magnification factor

$$\begin{aligned} M &= \frac{1}{2}[(\frac{ds^*}{ds})^2 - 1] && (3.39)\\ &= \epsilon_E + \frac{1}{2}\epsilon_E^2 && (3.40)\\ &= l^2\epsilon_{xx} + lm\epsilon_{xy} + ln\epsilon_{xz} + ml\epsilon_{yx} && (3.41)\\ &\quad + m^2\epsilon_{yy} + mn\epsilon_{yz} + nl\epsilon_{zx} + nm\epsilon_{zy} + n^2\epsilon_{zz} && (3.42) \end{aligned}$$

$$M = l^2\epsilon_{xx} + m^2\epsilon_{yy} + n^2\epsilon_{zz} + 2lm\epsilon_{xy} + 2ln\epsilon_{xz} + 2mn\epsilon_{yz}. \tag{3.43}$$

Although ϵ_E can be directly computed from (3.32), it is mathematically simpler to form the quantity M. For small ϵ_E, $\epsilon_E \cong$M. In M;

$$\epsilon_{xx} = \frac{\partial u}{\partial x} + \frac{1}{2}[(\frac{\partial u}{\partial x})^2 + (\frac{\partial v}{\partial x})^2 + (\frac{\partial w}{\partial x})^2] \tag{3.44}$$

$$\epsilon_{yy} = \frac{\partial v}{\partial y} + \frac{1}{2}[(\frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial y})^2 + (\frac{\partial w}{\partial y})^2] \tag{3.45}$$

$$\epsilon_{zz} = \frac{\partial w}{\partial z} + \frac{1}{2}[(\frac{\partial u}{\partial z})^2 + (\frac{\partial v}{\partial z})^2 + (\frac{\partial w}{\partial z})^2] \tag{3.46}$$

$$\epsilon_{xy} = \epsilon_{yx} = \frac{1}{2}(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} + \frac{\partial u}{\partial x}\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\frac{\partial v}{\partial y} + \frac{\partial w}{\partial x}\frac{\partial w}{\partial y}) \tag{3.47}$$

$$\epsilon_{xz} = \epsilon_{zx} = \frac{1}{2}(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} + \frac{\partial u}{\partial x}\frac{\partial u}{\partial z} + \frac{\partial v}{\partial x}\frac{\partial v}{\partial z} + \frac{\partial w}{\partial x}\frac{\partial w}{\partial z}) \tag{3.48}$$

$$\epsilon_{yz} = \epsilon_{zy} = \frac{1}{2}(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} + \frac{\partial u}{\partial y}\frac{\partial u}{\partial z} + \frac{\partial v}{\partial y}\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\frac{\partial w}{\partial z}) \tag{3.49}$$

are the finite strain-displacement relations and where

$$l = \frac{dx}{ds} \tag{3.50}$$

$$m = \frac{dy}{ds} \tag{3.51}$$

$$n = \frac{dz}{ds} \tag{3.52}$$

In small displacement theory, the quadratic terms in equations (3.44), (3.45), (3.46), (3.47), (3.48), (3.49) are neglected.

3.6 Torsion of Circular Cross-Section Bars

Consider a straight, constant-diameter circular bar, as shown in Figure 3.10. The loading consists of torques T, applied in opposite directions at the ends of the bar. The lateral or cylindrical surface is assumed to be stress free. The observable deformations of the bar are shown in Figure 3.11. At the beginning, the lines are parallel to the axis of the bar. After torsion, some helices appear to have become with the circumferential lines remaining circumferential, that is, no distortion of a cross section, at least on the circumferential surface of the bar. So here, we make two basic assumptions:

1) Any plane section which is perpendicular to the axis of the bar remains a plane section.

2) Each diameter of a section rotates with the same angle ϕ(x).

According to Figure 3.12, the displacements v and w of point P can be expressed in terms of the angle ϕ through which the cross section rotates as

$v(x,y,z) = y^* - y = rcos(\beta+\phi) - rcos\beta = y(cos\phi - 1) - zsin\phi$

$w(x,y,z) = z^* - z = rsin(\beta+\phi) - rsin\beta = z(cos\phi - 1) - ysin\phi$

The displacement in the direction of the axis of the bar is taken to be a constant u_0 that can be set equal to zero with no loss in generality. Using the linear strain-displacement relations then gives:

$$\epsilon_x = \frac{\partial u}{\partial x} = 0 \tag{3.53}$$

$$\epsilon_y = \frac{\partial v}{\partial y} = cos\phi - 1 \tag{3.54}$$

$$\epsilon_z = \frac{\partial w}{\partial z} = cos\phi - 1 \tag{3.55}$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = -ysin\phi\frac{d\phi}{dx} - zcos\phi\frac{d\phi}{dx} \tag{3.56}$$

$$\gamma_{yz} = \frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} = sin\phi - sin\phi = 0 \tag{3.57}$$

$$\gamma_{xz} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} = -zsin\phi\frac{d\phi}{dx} + ycos\phi\frac{d\phi}{dx} \tag{3.58}$$

If we make the following assumptions; $\frac{d\phi}{dx}$ is small, cos$\phi\approx$1, sin$\phi\approx\phi$ and that products of ϕ and $\frac{d\phi}{dx}$ can be neglected, we find:

$$\epsilon_x = \epsilon_y = \epsilon_z = \gamma_{yz} = 0 \tag{3.59}$$

$$\gamma_{xy} = -z\frac{d\phi}{dx} \tag{3.60}$$

$$\gamma_{xz} = y\frac{d\phi}{dx} \tag{3.61}$$

If we assume that the material is linear and isotropic so that

$$\tau_{xy} = G\gamma_{xy} = -Gz\frac{d\phi}{dx} \tag{3.62}$$

$$\tau_{xz} = G\gamma_{xz} = Gy\frac{d\phi}{dx} \tag{3.63}$$

According to Figure 3.13a, it is seen that the torque can be represented in terms of the stresses as

$$T = \int_{Area}(y\tau_{xz} - z\tau_{xy})dA \tag{3.64}$$

$$T = G\frac{d\phi}{dx}\int_{Area}(x^2 + y^2)dA = G\frac{d\phi}{dx}\int_{Area} r^2 dA = JG\frac{d\phi}{dx} \tag{3.65}$$

or

$$T = JG\frac{d\phi}{dx} \tag{3.66}$$

where $J = \frac{\pi a^4}{2}$ is the polar moment of the area of the circular cross section. (3.66) is the basic force-displacement relation for the elementary torsion problem. Eliminating $G\frac{d\phi}{dx}$ between (3.62), (3.63) and (3.66), the stresses can be expressed in terms of the torque T as

$$\tau_{xy} = -\frac{Tz}{J} \tag{3.67}$$

and

$$\tau_{xz} = \frac{Ty}{J} \tag{3.68}$$

These stresses are shown at an arbitrary location in the cross section in Figure 3.13b. The stress vector can be expressed as

$$t = -j\frac{Tz}{J} + k\frac{Ty}{J} = (-jz + ky)\frac{T}{J} \tag{3.69}$$

which, with z=rsinβ and r=cosβ, becomes

$$t = (-jsin\beta + kcos\beta)\frac{Tr}{J} \tag{3.70}$$

The vector in the brackets is the unit vector e_β in the direction shown. So, the direction of the shear stress is everywhere tangent to the circle at that location, and with $t = \tau e_\beta$ it follows that the shear stress is given by

$$\tau = \frac{Tr}{J} \tag{3.71}$$

for the case of the torsion of a circular bar.

In summary, given the torque transmitted, the stresses are determined by the (3.71) with the observable angle of twist ϕ obtained by integrating the force displacement relation T=JG$\frac{d\phi}{dx}$.

3.7 Torsion of a Single Thin-Walled Tube

The problem of the torsion of a single thin walled tube has an exact solution when the tube is a constant thickness circular tube. The stresses in Figure:3.14a are given by (3.71) with $J = \pi(a^4 - b^4)/2$. The corresponding torque-rate of twist equation is $T = JG\frac{d\phi}{dx}$. When the thickness t=a-b is small compared to the average radius $R = (a + b)/2$, the average shear stress in the tube can be computed according to $t_{avg} = TR/J$ (Figure:3.14b). Generic formulation for the torsion of a single tube can be formulated on an arbitrary cross section as shown in Figure:3.15a. In this model, the wall thickness, which is small compared to the tube diameter, can vary with position around the perimeter of the tube but not along the axis of the tube. The wall thickness, which is small compared to the tube diameter, can vary with position around the perimeter of the tube but not along the axis of the tube. Also the area and orientation of the cross section are constant along the tube, that is, there is no initial twist of the tube. The equilibrium equation is based on consideration of a typical segment, as shown in Figure:3.15b. At any point s around the periphery of the tube the shear stress is constant across the thickness. Summing forces in the axial or x-direction gives

$$\Sigma F_x = -\tau_1 t_1 \delta x + \tau_2 t_2 \delta x = 0 \tag{3.72}$$

from which $\tau_1 t_1 = \tau_2 t_2$. This means that the product of the shear stress and the thickness is a constant around the periphery of the tube. The product $q = \tau t$ as defined the shear flow, is constant around the periphery. According to Figure: 3.16a, it is seen that the total torque T is

$$T = \oint r\ df = \oint rq\ ds = q \oint r \quad ds \tag{3.73}$$

where since the shear flow is constant it can be brought outside the integral. According to Figure:3.16b :

$$T = q \oint (\rho cos\theta) ds = q \oint \rho(cos\theta ds) = q \oint 2dA = 2A_0 q \tag{3.74}$$

where A_0 is the area with the midline of the tube. Since $q = \tau t$, this relation can be written as :

$$\tau = \frac{T}{2A_0 t} \tag{3.75}$$

Equation (3.75) verifies that the maximum shear stress occurs at the point around the periphery of the tube where the thickness is a minimum. The constant character of the shear flow, which is always tangent to the tube is shown in Figure:3.17.

Chapter 4

COMPUTATIONAL TECHNIQUES FOR NANOSCALE SIMULATIONS

Until about 40 years ago, researchers computed the thermodynamic properties of interacting, bulk condensed-matter systems with analytical approximation methods for infinite systems. These analytical methods were valid only in the weakly interacting system limit and the approximations had to be carried out numerically beyond a few orders. In recent times, a new kind of approximation scheme based on exact numerical computation of properties of a finite-sample system has become the most common approach to studying interacting condensed-matter systems. Molecular dynamics (MD) refers to the situation where the motion of atoms or molecules have treated in approximate finite difference equations of Newtonian mechanics. Except when dealing with very light atoms and very low temperatures, the use of classical mechanics is well justified, [1].

Until about 20 years ago, MD computations primarily used simplistic pair potentials to describe inert gases in condensed-matter systems or the materials that tend to form hexagonal closed packing structures. A slow transition to describe dynamics of more complex condensed-matter systems such as metals or semiconductors with explicit or implicit many-body force-field functions began with embedded-atom-method type of potentials for metals and bond-order type potentials for semiconductors. Based on the variations of these three types potentials, researchers have proposed and used a wide variety of force-field functions in classical MD simulations. Many of the potentials are expected to work well in the regimes of physical parameters in which they were constructed in the first place, [6].

In recent years, several accurate quantum MD schemes have computed the forces between atoms at each time step with quantum mechanical calculations within the Born-Oppenheimer approximation. The dynamic motion for ionic positions are still governed by Newtonian or Hamiltonian mechanics and described by MD. The most widely known and accurate scheme is the Car-Parrinello MD method which describes the electronic states and

atomic forces using ab initio density functional method (usually within the local density approximation). Although we can now perform such ab initio MD simulations for systems consisting of a few hundred atoms, for a vast range of system sizes such calculations start to stretch the limits of present-day computational resources. In the intermediate regimes between large-scale classical MD and quantum Car-Parrinello MD methods, semiempirical quantum simulation approaches cover an important system size range where classical potentials are not accurate enough and ab initio computations are not feasible. The tight-binding molecular dynamics (TBMD) approach thus provides an important bridge between the accurate ab initio quantum MD and classical MD methods, [1].

In computational nanotechnology research, these three simulation methods can be used in a complementary manner to improve computational accuracy and efficiency. Based on experimental observations or theoretical dynamic and structure simulations, we can first investigate a nanosystems atomic structure. After we finalize the nanoscale systems configurations, we can investigate its electronic behaviors through static ab initio electronic energy minimization schemes or through studies of the systems quantum conductance behavior. Investigating the ab initio electronic structure provides highly accurate information about not only the systems thermodynamic minimum energy configurations but also the chemical reactions and charge transfers that occur when two nanoscale systems are brought together or taken apart. Studies of transport behavior are important in designing nanodevices, where the operating characteristics are usually determined by electronic, thermal, acoustic, or chemical signal transfer through the system, [1], [18], [19].

4.1 Classical molecular dynamics

In earlier days, the structural, mechanical, and thermal properties of interacting, bulk condensed matter systems were studied with analytical approximation methods for infinite systems. Numerical computer simulations of the finite sample systems have become more common recently because powerful computers to simulate nanoscale systems in full complexity are now available. Atomistic molecular dynamics (MD) refers most commonly to the situation where the motion of atoms or molecules is treated in approximate finite difference equations of Newtonian mechanics. Except when dealing with very light atoms and very low temperatures, the use of the classical MD method is well justified. In MD simulation,

the dynamic evolution of the system is governed by Newtons classical equation of motion,

$$d^2R_I/dt^2 = F_I = -dV/dR_I \tag{4.1}$$

which is derived from the classical Hamiltonian of the system,

$$H = \Sigma P_I^2/2M_I + V(R_I) \tag{4.2}$$

The atomic forces are derived as analytic derivatives of the interaction energy functions,

$$F_I(R_I) = -dV/dR_I \tag{4.3}$$

and are used to construct Newtons classical equations of motion which are second-order ordinary differential equations. In its global structure, a general MD code typically implements an algorithm to find a numerical solution of a set of coupled first-order ordinary differential equations given by the Hamiltonian formulation of Newtons second law. The equations of motion are numerically integrated forward in finite time steps using a predictor-corrector method [18].

The MD code for carbon based systems involves analytic many-body force field functions such as Tersoff-Brenner potentials [19] for C-C and C-H interactions. The Tersoff-Brenner potential is specially suited for carbon-based systems, such as diamond, graphite, fullerenes, and nanotubes [19], and has been used in a wide variety of scenarios with results in agreement with experimental observations. Currently, there is no universal analytic many-body force field function that works for all materials and in all scenarios. A major distinguishing feature of the Tersoff-Brenner potential for carbon-based systems is that short-range bonded interactions are reactive, so that chemical bonds can form and break during the course of a simulation. Therefore, compared to some other molecular dynamics codes, the neighbor list describing the environment of each atom includes only a few atoms and needs to be updated more frequently. The computational cost of the many-body bonded interactions is relatively high compared to the cost of similar methods with non-reactive interactions with simpler functional forms. As a result, the overall computational costs of both short-range interactions and long-range, non-bonding van der Waals interactions are roughly comparable. For large-scale atomistic modeling (from 10^5 upto 10^8 atoms), multiple processors are used for MD simulations, and the MD codes are generally parallelized [1], [18], [19].

4.2 Generalized tight-binding molecular dynamics

In recent years, several more accurate quantum molecular dynamics schemes have been developed in which the forces between atoms are computed at each time step via quantum mechanical calculations within the Born-Oppenheimer approximation. The dynamic motions for ionic positions are still governed by Newtonian or Hamiltonian mechanics and described by molecular dynamics. In the most general approach of full quantum mechanical description of materials, atoms are described as a collection of quantum mechanical particles, nuclei, and electrons, governed by the Schrödinger equation

$$H\Phi[R_I, r_i] = E_{tot}\Phi[R_I, r_i] \tag{4.4}$$

with the full quantum many-body Hamiltonian operator

$$H = \Sigma P_I^2/2M_I + \Sigma Z_I Z_J e^2/R_{IJ} + \Sigma p_i^2/2m_e + \Sigma e^2/r_{ij} - \Sigma Z_I e^2/|R_I - r_i| \tag{4.5}$$

where R_I and r are the coordinates of the nuclei and electrons. Using the Born-Oppenheimer approximation, the electronic degrees of freedom are assumed to follow adiabatically the corresponding nuclear positions, and the nuclei coordinates become classical variables. With this approximation, the full quantum many-body problem is reduced to a quantum many-electron problem

$$H[R_I]\Psi[r_i] = E_{el}\Psi[r_i] \tag{4.6}$$

where

$$H = \Sigma P_I^2/2M_I + H[R_I] \tag{4.7}$$

In the intermediate regimes, for up to few thousand atoms, the tight-binding molecular dynamics (TBMD) approach provides very good accuracy for both structural and mechanical characteristics. The computational efficiency of the tight-binding method derives from the fact that the quantum Hamiltonian of the system can be parameterized. Furthermore, the electronic structure information can be easily extracted from the tight-binding Hamiltonian, which in addition also contains the effects of angular forces in a natural way [7], [6], [18]

In the tight-binding method, further approximation simplifies the quantum many electron problem. We assume that the crystal potential is strong that when an ion captures an electron during its motion through the lattice, the electron remains at that site for a

long time before leaking, or tunneling, to the next ion site. During the capture interval, the electron orbits primarily around a single ion uninfluenced by other atoms, so that its state function is essentially that of an atomic orbital. Usually, the electron is tightly bound to its own atom. modulated by a Bloch wave-function phase factor for a periodic lattice. This ensures that an electron in a tight-binding level will be found, with equal probability, in any cell of the crystal, because its wave function changes only by the phase factor as one electron moves from one cell to another. The tight-binding method is computationally efficient because we can parameterize the Hamiltonian (4.7). Furthermore, we can easily extract the electronic structure information from the tight-binding Hamiltonian, which also contains the effects of angular forces in a natural way [9], [7].

TBMD methods advantage is that it can be used to find an energy-minimized structure of a nanoscale system under consideration without symmetry constraints. Sometimes a systems symmetry-unconstrained dynamic energy minimization can help us find the systems global energetic minimum, which is not easily conceptualized on the symmetry consideration alone. The parallelization of the TBMD code involves parallelizing the direct diagonalization (of the electronic Hamiltonian matrix) part as well as the MD part. Parallelizing a sparse symmetric matrix with many eigenvalues and eigenvectors is a complex bottleneck in the simulation of large intermediate-range system and requires new algorithms [19].

4.3 Ab initio simulation methods

Additionally, the ab-initio or first principles method is a simulation method to directly solve the complex quantum many-body Schrdinger equation using numerical algorithms. An ab-initio method provides a more accurate description of quantum mechanical behavior of materials properties even though the system size is currently limited to only about few hundred atoms. Current ab-initio simulation methods are based on a rigorous mathematical foundation of the density functional theory (DFT). This is derived from the fact that the ground state total electronic energy is a functional of the density of the system. It has been shown that the DFT can be reformulated as a single-electron problem with self-consistent effective potential, including all the exchange-correlation effects of electronics interactions:

$$H_I = p^2/2m_e + V_H(r) + V_{XC}[\rho(r)] + V_{ion-el}(r), \tag{4.8}$$

$$H_I \Psi_i(r) = \epsilon \Psi_i(r), i = 1, ..., N_{tot}, \tag{4.9}$$

$$\rho(r) = \Sigma |\Psi_i(r)|^2. \tag{4.10}$$

This single-electron Schrdinger equation is known as the Kohn-Sham equation, and the local density approximation (LDA) has been introduced to approximate the unknown effective exchange-correlation potential. This DFT-LDA method has been very successful in predicting materials properties without using any experimental inputs other than the identity of the constituent atoms. For practical applications, the DFT-LDA method has been implemented with a pseudopotential approximation and a plane wave (PW) basis expansion of single-electron wave functions. These approximations reduce the electronic structure problem to a self-consistent matrix diagonalization problem [18].

The TBMD method described earlier is another quantum mechanical simulation method based on the linear combination of atomic orbital approximations to describe the quantum mechanical electronic wave functions. Because of the simple basis expansion using atomic orbitals, the TBMD method is approximately 1,000 times more efficient than the ab initio method. However, the ab initio method provides a more accurate description of quantum mechanical behavior of materials properties even though it limits system size to a few hundred atoms. From this viewpoint MD, TBMD and ab initio methods form a complementary set of simulation tools to study diverse atomic-scale processes in nanodevice modeling [1].

The three simulation methods described above each have their own advantages and are suitable for studies for a variety of properties of material systems. MD simulations have the least computational cost, followed by Tight Binding methods. Ab initio methods are the most costly among the three. MD simulations can study systems with up to millions of atoms. With well-fitted empirical potentials, MD simulations are quite suitable for studies of dynamical properties of large-scale systems, where the detailed electronic properties of systems are not always necessary. While DFT methods can provide higher accuracy for self-consistent electronic structures, the high computational cost limits them to systems up to hundreds of atoms currently. To take the full capacity of DFT methods, a careful choice of an appropriate sized system is necessary. Tight Binding methods lay in between MD simulations and DFT methods, as to the computational cost and accuracy, and are applicable for systems up to thousands of atoms. For moderate-sized systems, TB methods can provide quite accurate mechanical and electronic characteristics [1], [6].

4.4 Parallel Computing

Parallel processing as a method of having many small tasks solve one large problem has emerged as a key enabling technology in modern computing. The past several years have witnessed an everincreasing acceptance and adoption of parallel processing, both for high-performance scientific computing and for more "general purpose" applications, was a result of the demand for higher performance, lower cost, and sustained productivity. The acceptance has been facilitated by two major developments: massively parallel processors (MPPs) and the widespread use of distributed computing [13], [14].

MPPs are now the most powerful computers in the world. These machines combine a few hundred to a few thousand CPUs in a single large cabinet connected to hundreds of gigabytes of memory. MPPs offer enormous computational power and are used to solve computational grand challenge problems such as global climate modeling and drug design. As simulations become more realistic, the computational power required to produce them grows rapidly. Thus, researchers on the cutting edge turn to MPPs and parallel processing in order to get the most computational power possible. The second major development affecting scientific problem solving is distributed computing. Distributed computing is a process whereby a set of computers connected by a network are used collectively to solve a single large problem. As more and more organizations acquire highspeed local area networks interconnecting many general purpose workstations, the combined computational resources may exceed the power of a single high performance computer. In some cases, several MPPs have been combined using distributed computing to produce unequaled computational power [13], [14], [15].

In all parallel processing, data must be exchanged between cooperating tasks. Several paradigms have been tried including shared memory, parallelizing compilers, and message passing. The message passing model has become the paradigm of choice, from the perspective of the number and variety of multiprocessors that support it, as well as in terms of applications, languages, and software systems that use it. The Parallel Virtual Machine (PVM) system is used in this study. PVM uses the message passing model to allow programmers to exploit distributed computing across a wide variety of computer types, including MPPs. A key concept in PVM is that it makes a collection of computers appear as one large virtual machine, hence its name [14].

PVM is an integrated set of software tools and libraries that emulates a general purpose,

flexible, heterogeneous concurrent computing framework on interconnected computers of varied architecture. The overall objective of the PVM system is to enable such a collection of computers to be used cooperatively for concurrent or parallel computation. Detailed descriptions and discussions of the concepts, logistics, and methodologies involved in this network based computing process are contained in the User Guide of PVM. Briefly, the principles upon which PVM is based include the following:

1) User configured host pool: The application's computational tasks are executed on a set of machines that are selected by the user for a given run of the PVM program. Both singleCPU machines and hardware multiprocessors (including shared memory and distributed memory computers) may be part of the host pool. The host pool may be altered by adding and deleting machines during operation. (This is an important feature for fault tolerance).

2) Translucent access to hardware: Application programs either may view the hardware environment as an attributeless collection of virtual processing elements or may choose to exploit the capabilities of specific machines in the host pool by positioning certain computational tasks on the most appropriate computers.

3) Process based computation: The unit of parallelism in PVM is a task, an independent sequential thread of control that alternates between communication and computation. No process to processor mapping is implied or enforced by PVM; in particular, multiple tasks may be executed on a single processor.

4) Explicit messagepassing model: Collections of computational tasks, each performing a part of an application's workload using data, functional, or hybrid decomposition, cooperate by explicitly sending and receiving messages to one another. Message size is limited only by the amount of available memory.

5) Heterogeneity support: The PVM system supports heterogeneity in terms of machines, networks, and applications. With regard to message passing, PVM permits messages containing more than one datatype to be exchanged between machines having different data representations.

6) Multiprocessor support: PVM uses the native messagepassing facilities on multiprocessors to take advantage of the underlying hardware [13], [16].

The PVM system is composed of two parts. The first part is a daemon, called pvmd3

and sometimes abbreviated pvmd, that resides on all the computers making up the virtual machine. Pvmd3 is designed so that any user with a valid login can install this daemon on a machine. When a user wishes to run a PVM application, first virtual machine must be created by starting up PVM. The PVM application can then be started from a Unix prompt on any of the hosts. Multiple users can configure overlapping virtual machines, and each user can execute several PVM applications simultaneously. The second part of the system is a library of PVM interface routines. It contains a functionally complete repertoire of primitives that are needed for cooperation between tasks of an application. This library contains usercallable routines for message passing, spawning processes, coordinating tasks, and modifying the virtual machine. The PVM computing model is based on the notion that an application consists of several tasks. Each task is responsible for a part of the application's computational workload. Sometimes an application is parallelized along its functions; that is, each task performs a different function, for example, input, problem setup, solution, output, and display. This process is often called functional parallelism. A more common method of parallelizing an application is called data parallelism. In this method all the tasks are the same, but each one only knows and solves a small part of the data. This is also referred to as the SPMD (single program multiple data) model of computing. PVM supports either or a mixture of these methods. Depending on their functions, tasks may execute in parallel and may need to synchronize or exchange data, although this is not always the case. [13]

The PVM system currently supports C, C++ and Fortran languages. This set of language interfaces have been included based on the observation that the predominant majority of target applications are written in C and Fortran, with an emerging trend in experimenting with object based languages and methodologies. The C and C++ language bindings for the PVM user interface library are implemented as functions, following the general conventions used by most C systems, including Unix like operating systems. To elaborate, function arguments are a combination of value parameters and pointers as appropriate, and function result values indicate the outcome of the call. In addition, macro definitions are used for system constants, and global variables such as errno and pvm errno are the mechanism for discriminating between multiple possible outcomes. Application programs written in C and C++ access PVM library functions by linking against an archival library (libpvm3.a)

that is part of the standard distribution. Fortran language bindings are implemented as subroutines rather than as functions. This approach was taken because some compilers on the supported architectures would not reliably interface Fortran functions with C functions. One immediate implication of this is that an additional argument is introduced into each PVM library call for status results to be returned to the invoking program. Also, library routines for the placement and retrieval of typed data in message buffers are unified, with an additional parameter indicating the datatype. Apart from these differences (and the standard naming prefixes — pvm for C, and pvmf for Fortran), a one to one correspondence exists between the two language bindings. Fortran interfaces to PVM are implemented as library stubs that in turn invoke the corresponding C routines, after casting and/or dereferencing arguments as appropriate. Thus, Fortran applications are required to link against the stubs library (libfpvm3.a) as well as the C library [13].

All PVM tasks are identified by an integer task identifier (TID) . Messages are sent to and received from TID. Since TID must be unique across the entire virtual machine, they are supplied by the local pvmd and are not user chosen. Although PVM encodes information into each TID the user is expected to treat the tids as opaque integer identifiers. PVM contains several routines that return TID values so that the user application can identify other tasks in the system. There are applications where it is natural to think of a group of tasks while cases exists where a user would like to identify his tasks by the numbers 0-(p-1), where p is the number of tasks. PVM includes the concept of user named groups. When a task joins a group, it is assigned a unique "instance" number in that group. Instance numbers start at 0 and count up. In keeping with the PVM philosophy, the group functions are designed to be very general and transparent to the user. For example, any PVM task can join or leave any group at any time without having to inform any other task in the affected groups. Also, groups can overlap, and tasks can broadcast messages to groups of which they are not a member. To use any of the group functions, a program must be linked with libgpvm3.a. The general paradigm for application programming with PVM is as follows. A user writes one or more sequential programing C, C++, or Fortran 77 that contain embedded calls to the PVM library. Each program corresponds to a task making up the application. These programs are compiled for each architecture in the host pool, and the resulting object files are placed at a location accessible from machines in the host

pool. To execute an application, a user typically starts one copy of one task (usually the "master" or "initiating" task) by hand from a machine within the host pool. This process subsequently starts other PVM tasks, eventually resulting in a collection of active tasks that then compute locally and exchange messages with each other to solve the problem. Tasks interact through explicit message passing, identifying each other with a system assigned, opaque TID. [13]

In this study, we used an O(N) parallel tight binding molecular dynamics simulation code developed in [7]. In that study the linear scaling of the code is provided by using divide and conquer approach to carry out quantum calculations. The basic strategy for this method is to divide a large system into many subsystems, determining the electron density of each subsystem separately, and summing the corresponding contributions from all subsystems to obtain only from the electron density. In traditional TB approach, the Hamiltonian of the system is solved in frequency space (reciprocal space, k space) by direct matrix diagonalization. This causes $O(N^3)$ scaling with respect to the number of atoms. But in the linear scaling method, the Hamiltonian of the system is solved in real space by making the approximation that only the local environment contributes to the bonding of each atom. Since this approximation is made, for an O(N) tbmd algorithm, there is a very important parameter which is called cutoff radius. The cutoff radius determines which two atoms are in interaction with each other. But during the molecular dynamics simulation, since the coordinates of the atoms change, the atoms which are in interaction in an MD step may not be interaction with each other in the next MD step. So the interacting pairs must be checked whether they are in interaction or not. But this search is quite expensive, being of the order N^2, if no special algorithms are applied [7]. To speed up this process, either the linked cells algorithm or a Verlet list or both of them can be used [7]. Since divide and conquer algorithm for tbmd is used, it is inherently possible to make use of linked cells technique and apply parallelization algorithm [7].

The main cost of all computation in an O(N) tbmd simulation is spent for calculation of energy and forces.The approach followed in the study [7] is ; not to distribute all parts of tbmd simulation to avoid higher cost of the internode communication time. Instead of that, only calculation of energy and forces are distributed to processors, collecting the resulting forces, and doing the rest of calculation (non-bonding potentials, velocity rescaling, etc.)

on only one processor. By this way, it becomes possible to reduce the communication requirements and to obtain a better scaling to large number of processors. The parallel O(N) tbmd flowchart used in the study [7] is given below

Master Node, Sending all necessary information to slaves

Slave Node

Divide and Conquer Scheme

–Computing Repulsive Forces and Energy, Urep

–From i = 1 to (Number of Cells/Number of Processors)

—-Computing O(N) Hamiltonian and diagonalization

—-Internode communication; Computing Ebs

–Internode communication; Cal. Chemical Potential for the Whole System

–From i = 1 to (Number of Cells/Number of Processors)

—-Computing the Hellmann.Feynman Forces partially

—-Internode communication; Computing the Hellmann.Feynman Forces

End of Divide and Conquer Scheme

Computing Kinetic Energy, Rescaling Velocities to keep Temperature constant

Compute Total Energy per Atom, ET = Urep + Ebs + Ekin

MD Loop,MD Time Step ≡ 1

–Compute new positions ri(n+ 1)

–Internode communication; Updating Positions in each node

–Divide and Conquer Scheme

–Compute new forces F_i(n + 1) and accelerations a_i(n+ 1)

–Compute new velocities v_i(n+ 1)

–Computing Instantaneous Temperature, Kinetic Energy

–Rescaling Atomic velocities to Keep Temperature Constant

–Computing Pair Correlation Function and Radial Distribution Function

–Computing Bond Angle Distribution and Atomic Coordination Number

–Computing Bond Length Distribution

–Compute Total Energy, ET = Urep + Ebs + Ekin

–Saving Intermediate Configuration

–Drawing Atomic Structure (PovRay)

–mds = mds + 1

End of MD Loop, (n ≡ MD time step; i = 1, . . . ,Number of Atoms)

Chapter 5

RESULTS AND DISCUSSION

In this thesis, we report an O(N) parallel tight binding molecular dynamics simulation study of (10x10) structured carbon nanotubes (CNT) at temperatures 300 K, 600 K, 900 K, 1200 K with different layers (20, 40, 50, 60) and applied torsion to these CNTs for different twist angles ($45^o, 90^o, 120^o, 180^o, 360^o$). The tight-binding (TB) theory has been established as a good compromise between ab initio simulations and model potential ones, bridging the gap between them, either as far as the overall numerical efficiency and /or as far as the accuracy were concerned. Tight-binding molecular dynamics (TBMD) is a computational tool designed to run finite temperature MD simulations within the semiempirical tight-binding scheme. This technique can be used to simulate material systems at different conditions of temperature, pressure, etc., including materials at extreme thermodynamical conditions. The electronic structure of simulated system can be calculated by a TB Hamiltonian so that the quantum mechanical many-body nature of interatomic forces is naturaly taken into account. The traditional TB theory solves the Schrödinger equation by direct matrix diagonalization, which results in cubic scaling with respect to the number of atoms [$O(N^3)$]. The O(N) methods, on the other hand, make the approximation that only the local environment contributes to the bonding and hence the band energy of each atom. In this case the run time would be in linear scaling with respect to the number of atoms. Moreover, O(N) schemes can be efficiently parallelized through the use of message passing libraries. Message passing libraries such as PVM and MPI are making simulations possible on clusters of computers [9]. All the simulations presented here were carried out in canonical (N, V, T) ensemble. The Newton equations of motion were integrated using the velocity Verlet algorithm with a time step equal to 0.1 fs. To avoid inaccurate integration of the Newton equations of motion, the velocities of the constituent atoms were occasionally rescaled to maintain the temperature of the system at target value. Periodic boundary conditions were applied in the uniaxial direction. The details of parallel O(N) TBMD algorithm used in this

study can be found in [7], [8], [9] [17].

In this study, there are three different kinds of comparisons for a 10×10 piristine CNT. These are the effects of layer number, temperature and twist angle changes on the torsional response of CNT. In our study, torsion is applied to the structure at one of its ends while the other end is kept fixed. In Figure 5.1, temperature effect on torsional response of CNT is given. MD steps versus total energy changes of the system results are given for 20 layers, 360^o twist angle, 10×10 piristine CNT. The temperatures values are set at 300 K, 600 K, 900 K and 1200 K. As seen from Figure 5.1, there is no significant difference in the torsion response of CNT for the temperature conditions of 300 K and 600 K. For both of these conditions there are separated atoms from the body of CNT. These can be easily detected from the jumps in Figure 5.1. These jumps and the separated atoms can be clearly seen in Figures 5.7 and 5.8, respectively. For 900 K, there are also separated atoms, but the total system stabilizes at a higher energy value compared with the lower temperature values. Another thing that must be noted here is that, there is only one separated atom for 900 K. This is an expected behavior since the temperature increase makes the total structure more ductile and this ductility provides the system to keep its atoms together more strongly as torsion is applied to the CNT. The effect of temperature on the ductility of the system can be easily seen when the temperature is increased to 1200 K. At higher temperature, as expected, the system stabilizes at a higher energy value. But more important observation here is that there are no atoms separated from the body of the CNT. This can be seen in Figures 5.10 and 5.22a-b. This is a very clear result that shows how the temperature effects the ductility of the system. The jumps of energy graphs, the separated atoms and the deformations have been shown clearly for the given temperature values in Figures 5.19a-b, 5.20a-b, 5.21a-b, 5.22a-b, 5.7, 5.8, 5.9, 5.10, respectively. In Figure 5.2, the effect of twist angle to the final total energy of the whole system (20 layer, 300 K, 10x10 CNT) is given. As expected, total energy of the system increases as the twist angle increases since the strain enegy stored in the system increases. But unexpectedly, the total energy, for the case of 360^o twist angle, decreases to an energy value which is close to the total energy value for the case of 45^o twist angle. The main reason for this unexpected behavior can be explained as follows : In the case of 45^o twist angle, there is no separated atom in the structure due to the applied torsion. This can be seen in Figures 5.2, 5.11 and 5.23a-b. In Figure 5.11,

there is no jump in the total energy value after torsion is applied. Jumps in total energy graphs mean separation of atoms from the structure due to the bond breaking between atoms. Also, in Figures 5.23a-b, it is clearly seen that no atoms have been separated from the piristine structure of 10×10 CNT. Briefly, all the strain energy due to applied torsion is stored in the structure. But for the case of 360^o twist angle, there are two separated atoms in the structure because of the applied torsion. This can be easily detected in Figures 5.7 and 5.19a-b. In Figure 5.7 there are two jumps in the total energy graph. This means that at these moments of MD step, two atoms have separated from the structure. Naturaly and expectedly, the bonds around these atoms have been broken. Total energy of the system decreases because of these relaxations of atoms and bonds. These separated atoms can be seen in Figure 5.19a-b. Figure 5.3 is a good example to investigate the effect of number of layers on the torsional response of CNTs. In this graph for the same temperature and twist angle values, torsional response of CNT's with 20 layer and 40 layers are shown. The total energy of the 20 layered carbon nanotube system is greater than the total energy of the 40 layered carbon nanotube system. This is due to the stored strain energy in the system as for the same amount of applied torsion, a shorter system has more displacement than the longer system. From Figures 5.24a-b, 5.26a-b, 5.12 and 5.14 it is seen that for 20 layered carbon nanotube there are 7 atoms separated from the structure. But for 40 layered carbon nanotube there are 11 separated atoms. This layer effect can not be seen when the applied twist angle is increased to 360^o. In Figure 5.4, for the 20 and 60 layered CNTs, at the tempearture of 300 K, a twist angle of 360^o is applied. As opposed to the Figure 5.3, at torsion step of molecular dynamic simulation, the total energy of the longer system is at higher values with respect to the shorter system. In Figure 5.27a-b, there are 29 atoms seen separated from the structure of carbon nanotube. This is almost 15-fold increased relaxation for the case of 20 layered carbon nanotube. But 60 layered CNT system stabilizes at a higher energy value than the 20 layered system. This may also be caused by the fact that the number of atoms contained in the structure of 60 layer system is larger. In Figure 5.5, the twist angle effect on the torsional response of 50 layered carbon nanotube is given. As expected, total energy of the system increases as the applied torsion increases. In the case of 50 layered carbon nanotube, the increase in the energy value for stabilization of the structure for twist angle 360^o occurs. As explained above, the total energy of the

system does not decrease as in the case of 20 layered carbon nanotube for twist angle 360^o. This is seen in Figures 5.5, 5.16 and 5.17. In Figure 5.6, a 50 layered CNT with twist angle of 180^o is investigated for an increasing temperature in the case of applied torsion. Same torsional response characteristics are seen here with 20 layered CNT response to the torsion. In Figure 5.29, the temperature effect on bond length distribution is given for a 20 layered 360^o twist angle 10×10 carbon nanotube. The most significant thing seen from this figure is that, when the temperature is increased, the lenghts of the bonds get longer in the case of same amount of torsion and this effect explains why the structure does not tear apart at higher temperatures. This effect can also be easily seen in Figure 5.1. Since in this figure, there are jumps for the temperature values 300 K, 600 K and 900 K. As explained above, these jumps show the breaking-of the atoms from the carbon nanotube. These separated atoms can be seen in Figures 5.19a-b, 5.20a-b and 5.21a-b. In Figures 5.22a-b, the case for 1200 K and 360^o twist angle is shown. Some deformations do exist at sides of the tube but as it can also be detected from Figure 5.1, there is no separated atom. In Figure 5.30, the twist angle effect on bond length distribution is given for a 20 layered, 300 K constant temperature 10×10 carbon nanotube. When the twist angle increases from 45^o to 90^o, longer values for the bond lengths are obtained in the distribution function. But when the twist angle increases from 90^o to 120^o, distribution function unexpectedly narrows beyond to the values of twist angle 45^o. This is a significant physical torsion response of carbon nanotube. As the twist angle is increased from 120^o to 360^o, the distribution function barely widens beyond to the values of 45^o. In Figure 5.31, the layer effect on torsional response of carbon nanotubes is given for twist angle 90^o. As the number of layers increases, the deformation over the bonds due to the torsion decreases. This is an expected behaviour since as the length of the tube increases, the strain energy stored in the system due to torsion decreases. This can also be easily detected from Figure 5.3. But in Figures 5.32 and 5.4, we see the opposite behaviour. In these plots we can see that as the number of layers increases, the stored strain energy in the system due to torsion with a twist angle of 360^o also increases. This is a very dramatic effect of layer number on torsional response of carbon nanotube. The Figures 5.33 and 5.34 shows the twist angle effect and temperature effect on bond length distribution of a 50 layered carbon nanotube. In Figures 5.35 and 5.36, the initial bond length distribution and bond angle distribution functions are given

respectively. In Figure 5.37, the temperature effect on bond angle distribution is given. As expected, the distribution function graphs widen as the temperature values increase. Due to torsion with a twist angle of 360^o, for higher temperatures, deformations over the structure of carbon nanotube increase dramatically and this causes the angle distribution functions to widen. In Figure 5.38, the twist angle effect on bond angle distribution is given with the same interesting behaviour as in Figure 5.30. For the increase in the twist angle value from 45^o to 90^o, distribution function widens as expected. But after 90^o, as the twist angle value increases, the distribution function narrows. The same behaviour for torsional response has been seen in bond length distributions. For twist angle values greater than 90^o, the structure of 10×10, 20 layer carbon nanotube tends to be homogenous in the means of angle and length values up to 120^o of twist angle. As we increase the twist angle from 120^o to 360^o, the distribution function widens. In Figure 5.39, layer effect on bond angle distribution function is given.The observed behaviour is expected since for a longer structure, the distortion in the angles will be less than that of a shorter structure in the case of torsion according to continium elesticity theorem. This is what exactly is seen in this figure. But in Figure 5.40, we cannot observe the same behaviour. In this figure, for a longer structure, 60 layered carbon nanotube, the distribution function is wider than that of a shorter structure, 20 layered carbon nanotube. This is due to difference in twist angle values. The structure behaves differently for the twist angle values smaller than 90^o and larger than 90^o. In Figure 5.41 the twist angle effect on bond length distribution is seen for a 50 layered carbon nanotube at a temperature of 300 K. In Figure 5.42, temperature effect is seen on bond length distribution for a 50 layered carbon nanotube with a twist angle of 180^o.The same behaviour is seen here with the Figures 5.34 and 5.6. The effect of the increase in the temperature is not very significant for a 50 layered carbon nanotube at the given twist angle value.

From the continuum elasticity theorem, the torsion stiffness of a carbon nanotube is defined as

$$K = \frac{1}{L}\frac{d^2E}{d\theta^2} \tag{5.1}$$

where E is the total energy of the structure, L is the axial length and θ is the applied twist angle. Also from continuum elasticity theorem, the shear strain and the total energy due

to shear strain are defined as

$$\epsilon = \frac{R\theta}{L} \tag{5.2}$$

and

$$E = \frac{1}{2}G\int(\epsilon^2 dV), \tag{5.3}$$

respectively. V is the volume, R is the radius and G is the shear modulus of carbon nanotube. Doing the necessary simplifications, we can obtain the the torsion stiffness as

$$K = G(2\pi h)\frac{R^3}{L^3} \tag{5.4}$$

where h is the wall thickness of carbon nanotube and is equal to 1.42 Angstrom. Also, by writing the shear strain expression into the total energy expression and evaluating the integral, and making necessary modifications, we can obtain an analytical expression for shear modulus as

$$G = \frac{2EL^2}{R^2\theta^2(V_2 - V_1)} \tag{5.5}$$

Therefore, we can define the torsion stiffness

$$K = CG \tag{5.6}$$

where C is a constant given by

$$C = \frac{2\pi hR^3}{L^2}. \tag{5.7}$$

For the above expressions, the unit of shear modulus is $[eV/radian^2 Angstrom^3]$ which is equal to TPa and the unit of torsion stiffness is $[eV/radian^2 Angstrom]$. In Figure 5.43, the effect of twist angle on the shear modulus of 10×10, 20 layered carbon nanotube at temperature of 300 K is shown. Shear modulus is a coefficient of elasticity for a shearing force. It is defined as the ratio of shear stress to the displacement per unit sample length [10]. Briefly, it is a material property defining the resistance to shearing stresses. In this study, this shearing force is the applied torsion. From the figure it is clearly seen that, as the twist angle increases, the resistance of the structure to shearing stresses decreases. In Figure 5.44, the effect of temperature on the shear modulus of 10×10, 20 layered carbon nanotube with the twist angle of 360^o is shown. As the temperature increases from 300 K to 600 K, shear modulus of the structure increases unexpectedly. But after this temperature, resistance of the structure decreases against shearing forces as the temperature increases. In

Figures 5.45 and 5.46, twist angle and temperature effects on the torsion stiffness of 10×10, 20 layered carbon nanotube is given with respect to different twist angles and different temperature values. Since torsion stiffness is obtained from shear modulus by multiplying it with a constant which is obtained from radius and axial length of the carbon nanotube, Figures 5.45 and 5.46 have the same characteristics as the Figures 5.43 and 5.44.

Finally, in Figures 5.47 and 5.48, efficiency and speedup tendency of the tbmd simulation O(N) algorithm is given in the case of parallel execution. In Figure 5.47, as the number of processors increases, the efficiency decreases. Also in Figure 5.48, the speedup value increases upto 4 processors. After 4 processors speedup value starts to decrease. The main reason for this decrease is the increase of communication time spend between processors. Because of this communication time, speedup can not increase linearly.

Chapter 6

CONCLUSION

In this study, O(N) tight binding molecular dynamics simulation method of Dereli and Özdoğan [6] has been applied to a computational physics problem on the parallel computation platform at Koç University by using PVM library. The mechanical properties of (n,m)=(10,10) SWNTs with 20, 40, 50 and 60 layers have been computed for different magnitudes of torsional load and temperature conditions. Shear modulus and torsion stiffness of the 20 layered SWNT have been computed. Efficiency and speedup effect of parallel programming is presented. In conclusion, the following observations can be made : Firstly, the torsion stiffness of the carbon nanotube we studied decreases as the twist angle increases. This is the expected behavior. The expectations from the continuum elastic theory are not met for two other cases. The torsion stiffness increases with increasing temperature up to a point (shown and explained in Chapter 5) and then it decreases with further increase in temperature. This significant effect needs a further study. Moreover, the torsional response of carbon nanotubes as given by the bond-angle and bond-length distributions exhibit distinct behavior below and above a specific twist angle value which is about 90^o degrees for the SWNTs we studied. The precise value seems to be insensitive to the number of layers. This effect also needs to be further investigated and should be understood. Finally we would like to comment on parallel computation. From the efficiency and speedup graphs it is seen that parallel computation is advantageous in computational time with 4 processors only. Increasing the number of processors didn't give any improvement. As a future project, using the same O(N) TBMD algorithm, the response characteristics of carbon nanotubes in the case of bending, complex bending (combination of torsion and bending) and buckling can be investigated.

BIBLIOGRAPHY

[1] B. Bhushan (Editor), **Handbook of Nanotechnology**, (Springer, 2003)

[2] M.S. Dresselhaus, G. Dresselhaus, P.C. Eklund, **Science of Fullerenes and Carbon Nanotubes**, (Academic Press, 1996)

[3] R. Saito, G. Dresselhaus, M.S. Dresselhaus, **Physical Properties of Carbon Nanotubes**, (Imperial College Press, 1998)

[4] www.wag.caltech.edu/foresight/foresight_2.html

[5] www.pa.msu.edu./cmp/csc/ntproperties/equilibriumstructure.html

[6] D.C. Rapaport, **The Art of Molecular Dynamics Simulation**, (Cambridge University Press, 1997)

[7] C. Özdoğan, G. Dereli, T. Cağın, O(N) parallel tight binding molecular dynamics simulation of carbon nanotubes, Computer Physics Communications **148**, (2002), 188-205.

[8] G.Dereli, C. Özdoğan, Structural stability and energetics of single-walled carbon nanotubes under uniaxial strain, Physical Review **B67**, (2003),035416.

[9] G.Dereli, C. Özdoğan, O(N) algorithms in tight binding molecular dynamics simulations of the electronic structure of carbon nanotubes, Physical Review **B67**, (2003), 035415.

[10] Robert D. Cook, Waren C. Young, **Advanced Mechanics of Materials**, (Prentice Hall, Second Edition, 1999)

[11] William B. Bicford, **Advanced Mechanics of Materials**, (Addison Wesley, 1998)

[12] Arthur Boresi, Richard Schmidt, Omar Sidebottom, **Advanced Mechanics of Materials**, (John Wiley&Sons, Fifth Edition, 1993)

[13] A. Geist, A. Beguelin, W. Jiang, R. Manchek, V. Sunderam, **PVM Users' Guide and Tutorial for Networked Parallel Computing** , (The MIT Press, 1996)

[14] D. Culler, J. Singh, A. Gupta, **Parallel Computer Architecture : A Hardware/Software Approach** , (Morgan Kaufmann Publishers)

[15] A. Grama, A. Gupta, G. Karypis, V. Kumar, **Introduction to Parallel Computing** (Addison Wesley, Second Edition, 2003)

[16] J. Dongarra, B. Tourancheau (Editors),**Environments and Tools for Parallel Scientific Computing**, (Amsterdam ; New York : North-Holland, 1993)

[17] W. Yu, W.X. Xi, N. Xianggui, Atomistic simulation of the torsion deformation of carbon nanotubes, Modelling and Simulation in Materials Science and Engineering, **12**, (2004), 1099-107.

[18] D. Srivastava, M. Menon, K Cho, Computational Nanotechnology with Carbon Nanotubes and Fullerenes, IEEE Computing in Science and Engineering, **1521 − 9615**, (2001), 42-55.

[19] D. Srivastava, S.T. Barnard, Molecular dynamics simulation of large-scale carbon nanotubes on a shared-memory architecture, SC97 Technical Paper, 1997 ACM 0-89791-985-8/97/0011

VITA

UFUK PARALI was born in Delice,Kırıkkale, Turkey on July 20, 1979. He received his B.Sc. degree in Aeronautical Engineering and in Electronics & Telecommunication Engineering from Istanbul Technical University, Istanbul, Turkey in 2002. He worked as an R&D engineer in Northern Electric Telecommunication A.S. in 2003 for eight months. From August 2003 to August 2005, he attended Computational Sciences and Engineering Graduate Program in Koc University, Istanbul, Turkey. He worked as a teaching and research assistant in the Department of Physics.