A Computational Study of Carbon Nanotubes Embedded In Silicon

by

Jaren Quinn Norris

Submitted to Brigham Young University in partial fulfillment of graduation requirements for University Honors

Department of Physics and Astronomy

Brigham Young University

August 2011

Advisor: Bret C. Hess

Signature: _____

Honors Representative: Steve Turley

Signature: _____

ABSTRACT

A Computational Study of Carbon Nanotubes
Embedded In Silicon

Jaren Quinn Norris

Department of Physics and Astronomy

Bachelor of Science

Inspired by recent experimental success in coating carbon nanotubes with silicon, I have computationally modeled carbon nanotubes embedded in silicon using density functional theory. I have characterized the binding energy and electronic structure of several silicon-nanotube systems with different nanotube radii and orientations in the silicon lattice.

ACKNOWLEDGMENTS

I would like to give special thanks Dr. Bret Hess for suggesting this project and for two years of patient mentoring. I also wish to thank Daniel Jensen for guidance in developing my programs, and Tim Hecht for early collaboration on this project and for solving VASP compiling issues. In addition, I want to thank BYU Fulton Supercomputing Lab for access to the computing resources that made this project possible.

Contents

List of Figures

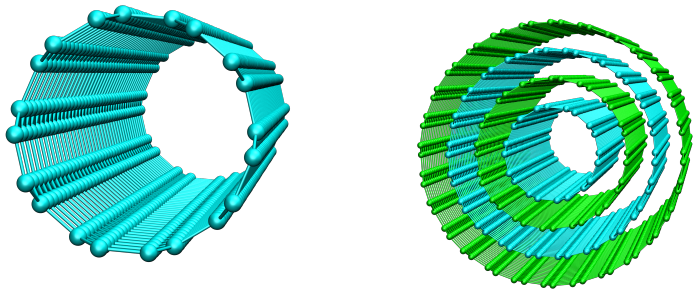
Chapter 1

Background and Significance

1.1 What Are Carbon Nanotubes?

Carbon nanotubes are tubes of carbon atoms with diameters on the order of a nanometer (10^{-9} meters). They can be anywhere from a few tenths of a nanometer to eighteen centimeters long [1]. They are classified with buckyballs as fullerenes, one of the allotropes, or types, of carbon molecules. Other allotropes of carbon are diamond, graphite and amorphous carbon. Carbon nanotubes are found naturally in soot and smoke and have been created in laboratories using arc discharge [2], laser ablation [3], and chemical vapor deposition [4]. They can have either one layer of carbon atoms (single-walled Fig. 1.1(a)), or have multiple tubes inside of tubes (multi-walled Fig. 1.1(b)).

While there are entire books devoted to carbon nanotubes, I wish give just an introduction to the geometry, discovery, properties, and applications of carbon nanotubes.



(a) Single-walled (b) Muti-walled

Figure 1.1 Representations of single-walled and multi-walled carbon nanotubes. The spheres represent carbon atoms and the small cylinders represent the bonds between carbon atoms.

1.1.1 Geometry

One of the best ways to think about single-walled carbon nanotubes is to think of them as rolled up sheets of graphene. Graphene is a two dimensional hexagonal array of carbon atoms(see Fig. 1.2). Graphite is made up of sheets of graphene stacked on top of each other and held together by Van der Waals forces, the attractive forces between the dynamic dipoles in the different molecules. Although this view of carbon nanotubes can be very useful, it is important to note that this is not how carbon nanotube are actually created.

The hexagonal structure of graphene can be generated by placing carbon atoms at linear combinations of the graphene lattice vectors $\mathbf{a_1}$ and $\mathbf{a_2}$(see Fig. 1.3). For a more complete explanation on generating crystal structures see Sec. 1.3.1. A single-walled nanotube can be classified by the geometry of the rectangular sheet of graphene that could be used to create it. One side of the rectangular sheet is defined by a linear combination of the two graphene lattice vectors $n\mathbf{a_1} + m\mathbf{a_2}$, called the chiral vector $\mathbf{C}$. The other side of the rectangle is defined by the shortest linear

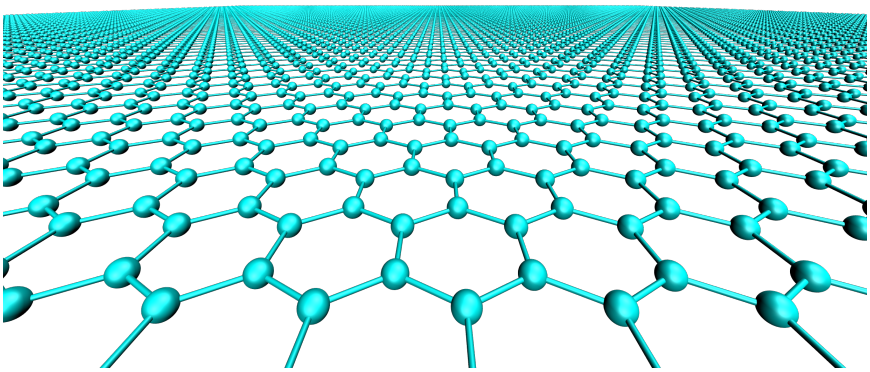


Figure 1.2 A large sheet of graphene.

combination of the graphene lattice vectors that is perpendicular to the chiral vector, and is called the translation vector **T**. The nanotube unit cell is formed by curling the chiral vector around an axis parallel to the translational vector. Longer carbon nanotubes can be made by repeating this unit cell until the desired length is reached. Multi-walled carbon nanotubes can be made by putting several single-walled carbon nanotubes inside each other. This structure makes carbon nanotubes effectively one dimensional.

As there is only one possible translation vector for each chiral vector, the chiral vector is used to classify carbon nanotubes. The component of each of the lattice vectors is listed as an ordered pair (n, m), called the chirality of the nanotube, which along with the length uniquely defines the carbon nanotube geometry. Knowing the chirality allows us to calculate the particular nanotube radius Eq. (1.1), repeat distance Eq. (1.2), the number of atoms in a unit cell Eq. (1.3), and to classify it as a metal or semiconductor Eq. (1.4).

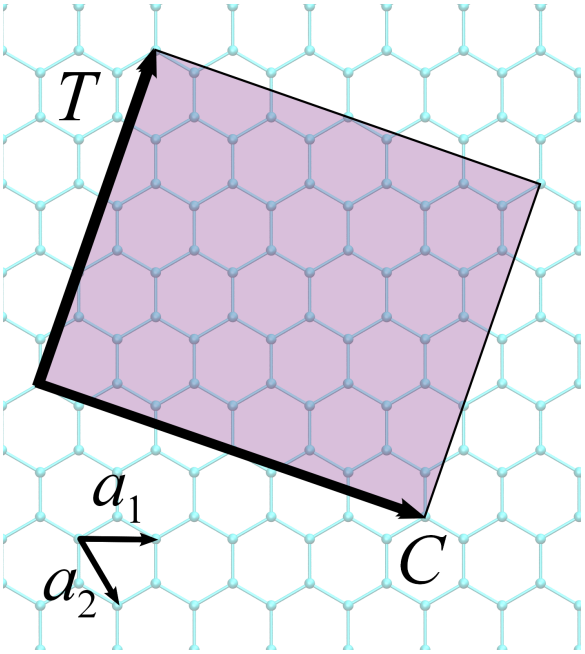


Figure 1.3 The shaded area is an example of a rectangular sheet of graphene that could be used to create a carbon nanotube. The graphene lattice vectors $\mathbf{a_1}$ and $\mathbf{a_2}$, the chiral vector $\mathbf{C} = n\mathbf{a_1} + m\mathbf{a_2}$ and the translation vector $\mathbf{T}$ have been labeled.

$$\text{nanotube radius} = \frac{a_0}{2\pi}\sqrt{n^2 + nm + m^2} \tag{1.1}$$

$$\text{nanotube repeat distance} = a_0\frac{\sqrt{3\,(n^2 + nm + m^2)}}{\gcd(n,m)R} \tag{1.2}$$

$$\text{number of atoms per unit cell} = \frac{2(n^2 + nm + m^2)}{\gcd(n,m)R} \tag{1.3}$$

Where $R = 3$ if $(n - m)/3$ is an integer, and $R = 1$ otherwise.

$$\text{nanotube conductivity} = \begin{cases} \text{metallic} & \text{if } m = n \\ \text{semiconducting with a very small band gap} & \text{if } (n-m)/3 \text{ is an integer} \\ \text{semiconducting} & \text{otherwise} \end{cases} \tag{1.4}$$

There are two special cases that are named for the shape of the ends of the nanotube unit cell. Zig-zag nanotubes Fig. 1.4 (a) have chiralities where $m = 0$ and armchair nanotubes Fig. 1.4 (b) have chiralities where $m = n$. Both zig-zag and armchair nanotubes are commonly studied in theory because they have the smallest unit cells for a given radius and because armchair nanotubes are the only metallic nanotubes.

(a) zig-zag (b) armchair

Figure 1.4 Examples of commonly studied carbon nanotube types named for the shapes of the edges of their unit cells.

1.1.2 Discovery

Carbon nanotubes were originally discovered in 1952 by Radushkevich and Lukyanovich in Russia using recently invented TEM [5]. They were observed by other scientists for the next 40 years [5], but did not begin to gain popularity until 1991, when Sumio Iijima at NEC Corporation discovered multi-walled carbon nanotubes produced during arc discharge evaporation of graphite rods [2]. The popularity of this observation

can be attributed to the prediction three months later by Mintmire, Dunlap, and White at the Naval Research Laboratory in Washington DC that carbon nanotubes should have remarkable electronic properties [6]. Since 1991 the number of articles published per year on carbon nanotubes has grown exponentially (Fig. 1.5).

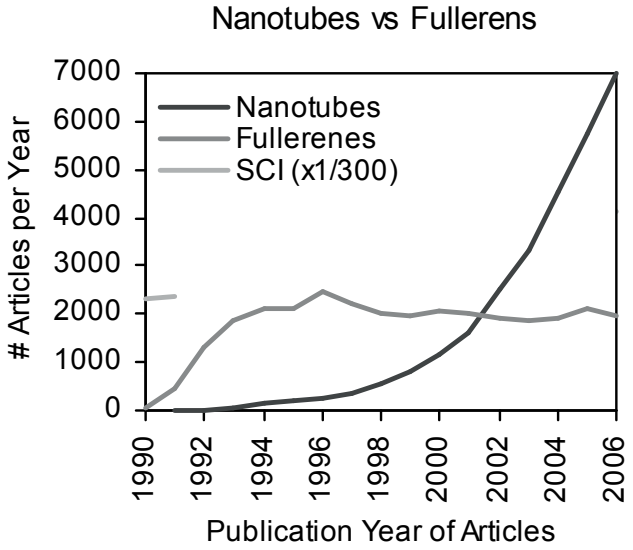


Figure 1.5 The growth in the number of articles and conference addresses on nanotubes, fullerenes and all subjects combined as recorded in the Science Citation Index [7]

1.1.3 Properties

Carbon nanotubes have been called "the new wonder-material of the future" [8] due to their remarkable properties and have been a major catalyst in the growth of the fields of nanotechnonlogy and nanoengineering. While they have been shown to have many novel properties, such as the ability to form superhydrophobic films [9], I will just introduce their mechanical, electrical and thermal properties.

Mechanical

The mechanical properties of carbon nanotubes are primarily the result of the strong covalent bonds holding the carbon atoms together. In terms of chemistry orbital hybridization, the nanotubes are bound by sp^2 bonds. These bonds are formed when one s and two p orbitals mix to from three hybridized sp^2 orbitals. These orbitals are lobe shaped, similar to p orbitals except that one side is much larger than the other. In addition, rather than lining up orthogonal to each other like with p orbitals, the axis of rotation for the lobes lie in a plane with 120 degrees between them Fig. 1.6. The sp^2 bonds account for the strength and hexagonal shape of these bonds.

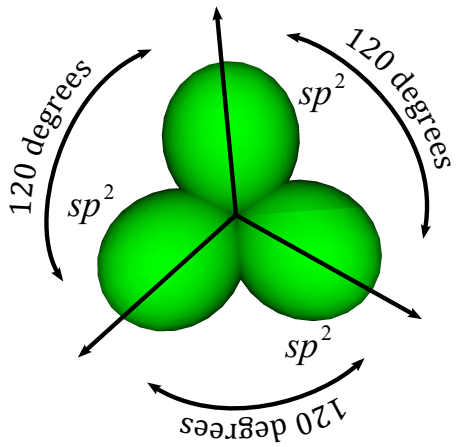


Figure 1.6 The sp^2 hybridized orbitals are responsible for many properties of carbon nanotubes.

It is often stated that carbon nanotubes are over 100 times stronger than steel, but it is important to know how that strength is defined. Two common mechanical properties are are Young's modulus, or tensile stiffness, and tensile strength. Young's modulus is the ratio of stress, the force per unit area, to strain, how much an object changes shape, Eq. (1.5). Basically it is a measure of how stretchy or squishy a material is.

$$\text{Young's Modulus} = \frac{\text{Stress}}{\text{Strain}} = \frac{\text{(Force)/(Initial Area)}}{\text{(Change in Length)/(Initial Length)}} \tag{1.5}$$

Theoretical models predict that carbon nanotubes should have a Young's modulus on the order of 1 terapascal(TPa) [10], five times that of steel and comparable to diamond. To put this in perspective you would have to pull on a carbon nanotube with a pressure 10 million times greater than the atmosphere, or 10 thousand times greater than the pressure at the deepest part of the ocean to get it to stretch significantly. The experimental values vary from 0.4 to 4.5 TPa with the most recent methods putting the Young's modulus at around 1.2 TPa [10]. This wide variation is the result of current difficulties in working at the nanometer scale. It is important to note that this Young's modulus only applies to forces applied along the axis of the tube, as if you were pulling on it like a rope. Carbon nanotubes are 10 to 100 times softer in the radial direction [11] and can be deformed by relatively weak Van der Waals forces [12].

One of the most exciting properties of carbon nanotubes is their tensile strength. Tensile strength is the maximum stress a material can withstand before its breaks. It is experimentally measured by pulling on a material and measuring the strain in response to applied force. Theoretical models predict that single wall carbon nanotubes should have a tensile strength on the order of 100 gigapascals(GPa) [13], or about 100 times that of steel. Experiments have measured tensile strengths around 30 GPa for single-walled nanotubes [14] and up to 63 GPa for multi-walled nanotubes [15]. nanotubes. The difference between the experimental and theoretical values is most likely due to defects in the nanotubes. Combining the large tensile strength with the low density of carbon nanotubes produces a strength-to-weight ratio 250 times greater than steel. This high tensile strength is aided by the fact that carbon

nanotubes spontaneously align and bundle together because of Van der Waals forces. Once again, it is important to note that tensile only applies to pulling forces applied along the axis of the tube. Carbon nanotubes buckle under compression around 1.7 GPa [16].

Electrical

In addition, carbon nanotubes display remarkable electrical properties. As stated in Sec. 1.1.1, the conductivity of a single walled nanotube depends only on chirality. This means that nanotubes of similar radius can be conducting, semi-conducting or insulating just by changes in geometry. Since the conductance of a carbon nanotube is dependent upon geometry, the conductance of a carbon nanotube can be temporarily changed by applying a force to deform the nanotube [17]. Furthermore, the large number of possible combinations, combined with doping can create band gaps that can be tuned to whatever is required [18]. It has also been shown that carbon nanotubes can form single molecule diodes [19] and room temperature transistors capable of switching on a single electron [20]. These properties make carbon nanotubes, a complete set of electronic building blocks, a candidate for miniaturizing any larger scale silicon based circuits.

Two other prominent electrical properties of carbon nanotubes are their potential current densities and carrier mobilities. Current density is the total electrical current divided by the cross sectional area through which that current flows. Metalized carbon nanotubes can carry current densities of 400 A/mm^2 [21], 100 times the typical current density for copper wire. This means a carbon nanotube bundle only 4 mm in diameter could replace a 4 cm diameter copper wire. Carrier mobility is defined by the average velocity of carriers, electrons or holes, divided by the electric field applied to produce

their motion. Carrier mobility characterizes how quickly electrons or holes can move when placed in an electric field and is important for making good semiconductor transistors. Experiments have measured carrier mobilities of carbon nanotubes at room temperature up to 100,000 cm^2/Vs [22], more than 70 times that of silicon.

Thermal

The thermal conductance of single-walled nanotubes at room temperature has been theoretically predicted to be near 6600 W/mK [23]. Experiments have measured their conductance near 3500 W/mK [24], roughly 10 times greater than good metallic conductors like copper. Like the mechanical properties of carbon nanotubes, this high value only applies to conductivity in the axial direction. The radial conductance of carbon nanotubes has been measure to be only 1.64 W/mK [25], close to the conductivity of glass.

1.1.4 Applications

The theoretical prediction of carbon nanotubes remarkable properties sparked inflated speculation that carbon nanotubes are the "the ultimate fiber" [26] and have "as much potential for revolutionizing our world as the last century's developments in computing and biotechnology" [27]. Proposed applications for carbon nanotubes include everything from micro robots to space elevators [28]. However, the majority of these ideal applications are years away as researchers are working to overcome challenges like detecting and eliminating nanotube defects, isolating nanotube chiralities, and nanoscale manipulation. The current method for detecting and eliminating defects is to generate a large number of devices and measure their properties. The ones that fail to perform are then know to have defects and are discarded [29]. Not

only does this manufacturing method fail to meet the reliability demands of industry, it reveals nothing about the type and character of the defects, or how to eliminate them in future devises.

Often an application requires carbon nanotubes of specific lengths and chiralities. There are several methods for sorting carbon nanotubes based on certain properties. Ultracentrifugation is capable of separating carbon nanotubes by their radii [30], but as mentioned earlier carbon nanotubes with similar radii can have vastly different electrical properties. Using column chromatography, carbon nanotubes can be divided into metallic and semiconducting with 90 and 95 percent purity respectfully [31]. This method is suitable for isolating nanoscale wires, but is unable to isolate semiconducting nanotubes with a particular band gap. By wrapping carbon nanotubes in DNA and using size exclusion chromatography, scientists have been able to separate carbon nanotubes by their lengths, with variations as small as 10 percent [32]. Using identified DNA tags and chromatography, 12 semiconduting chiralities have been isolated to purities between 70 and 90 percent [33]. While these methods are valuables steps in working towards isolating particular nantobues, none achieve the purities required for industrial implementation.

Carbon nanotubes are already being used in applications where specific chirality is not required, like adding carbon nanotubes to composite resins and plastics. Journals are filled with articles containing prototypes and laboratory demonstrations of carbon nanotube technology. I will introduce just two creative applications that take advantage of less sensational properties of carbon nanotubes.

In 2003 scientists at the University of Manchester, succeeded in replicating the "stickiness" of gecko toe pads using micrometer scale polymer pyramids. These pyramids where shown to have a macroscopic adhesion strength of 3 N/cm^2, close to

the 10 N/cm^2 for gecko toe pads [34]. Over the next few years, scientists were able to use carbon naotubes in the place of polymer pyramids and achieved macroscopic adhesion strengths of 36 N/cm^2 [35] and microscopic adhesion strengths 200 times that of of a gecko [36]. It was later shown that these dry tapes are electrically and thermally conductive [37] and display self-cleaning properties that retain 90 percent adhesion strength after being soiled and mechanically cleaned [38]. These tapes display a directionally-dependent adhesion and can easily be removed without leaving any residue by pulling the the right direction [35]. The stickiness of the gecko toe pads and these biomimetic "gecko" tapes come from Van der Waals forces. Both the pads and the tapes are covered with tiny bristle-like structures called satae Fig. 1.7. When pressed against a surface the pillars lay flat against the surface, creating a large area of close contact where Van der Waals forces can have the greatest effect. It is the sum of many tiny forces that gives gecko tape its incredible adhesion strength.

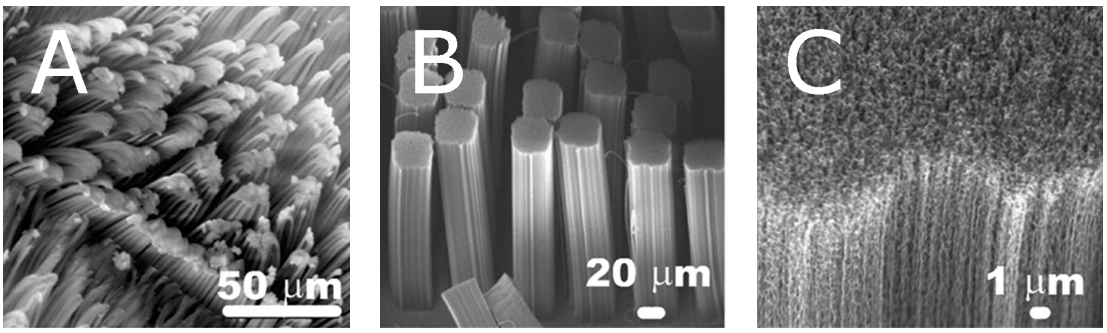


Figure 1.7 SEM images of A. The toe pad of a gecko, B. Biomimetic "gecko" tape showing micro patterned carbon nanotube bundles and C. Higher magnificaiton of B showing individual carbon nanotubes [38]

Another example of a novel application of carbon nanotubes is a transparent, flexible speaker made by scientists in China [39]. The speaker is made by stretching arrays of carbon nanotubes across a frame and applying a signal current across the array. When acoustically characterized, they were shown to produce a wide range

of frequencies at high volumes with little distortion. These arrays are 78 percent transparent, and can be bent in any direction, including a cylinder, creating a speaker capable of sending sound in specific directions. They can be stretched to twice their normal size with little loss in sound quality Fig. 1.8 These properties mean that a carbon nanotube speaker could be placed directly on top of a display to reduce the size of video devices. In addition, they do not contain any magnets and could be used in situations where the large magnets of traditional speakers cause interference or other problems.

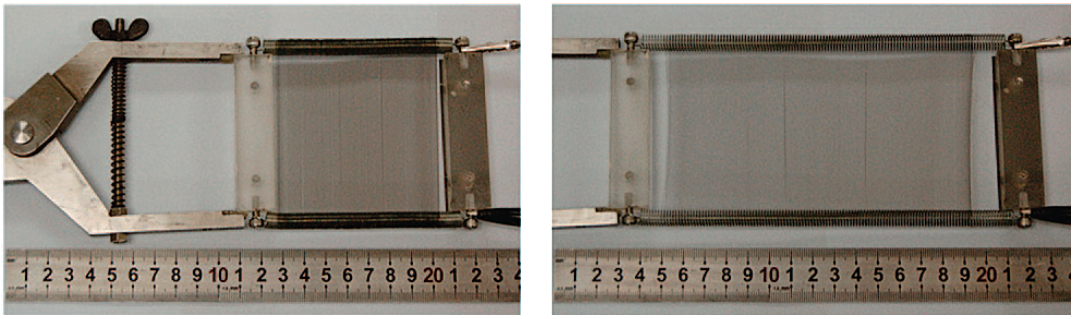


Figure 1.8 A novel carbon nanotube loudspeaker that can be stretched to stretched to twice its normal size with little change in sound quality. It produces sound using thermoacoustics. [39]

Carbon nanotube speakers produce sound the same way that lightning produces thunder. When the current flows through to nanotube array, the nanotubes heat up, which in turn heats the surrounding air, causing it to expand. Since carbon nanotubes have such a low heat capacity, they can change temperature rapidly enough to cause the air to expand and create the pressure waves needed for sound. This method of producing sound is called the thermoacoustic effect.

1.2 Motivation

The research published since 1991 has attempted to definitively quantify the extraordinary electrical, thermal and mechanical properties of carbon nanotubes. Often, experimental or theoretical methods have produced greatly differing values for a certain property [10] as concepts like cross-sectional area are difficult to define at the nanoscale [40]. As time has progressed, certain values and methods have gained wider acceptance, but more research is needed to understand why there are such large differences, and and to ensure the reproducibility of those results.

Recently at BYU, the Nanoscale Physics Group led by Dr. Davis developed a method that uses carbon nanotubes to create high aspect ratio MEMS devices, like the one shown in Fig. 1.9. The process involves growing carbon nanotube forests on a semiconductor substrate and then infiltrating the forests with silicon. These structures were shown to have the elastic properties of the infiltration material while retaining the conductivity of the carbon nanotubes [41]. The starting point of this study was to investigate how the electronic properties of both the carbon nanotubes and the silicon influence the electronic properties of the combined material. Additionally, I wanted to look at how the nanotube radius, direction of the nanotubes in the silicon lattice and the electronic properties of the nanotube influenced the electronic properties of the combined material

The majority of experimental studies of carbon nanotubes focus on micron scale fabrication and determining the physical properties of fabricated structures. On the other hand, most computational studies have focused on individual carbon nanotubes and their properties, as computational studies of larger structures are very computationally intensive. This study extends the scale of computational research, helping to bridge the gap between experimental and computational research.

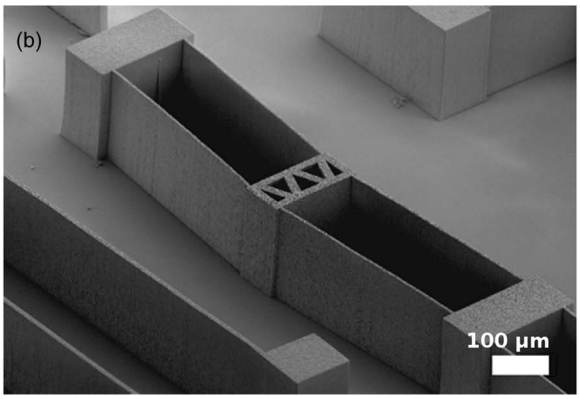


Figure 1.9 A high aspect-ratio microelectromechanical systems(MEMS) device produce by the BYU Nanoscale Physics Group [41].

1.3 Brief Introduction to Solid State Physics

This section is a basic overview of just a few principles of solid state physics. This section will give readers who are unfamiliar with solid state physics the background necessary to understand the topics discussed in subsequent chapters. Readers who are already familiar with these principles may wish to skip to Ch. 2.

1.3.1 Unit Cells

The unit cell is an essential concept in theoretical and computational solid state physics and one that is best explained by example. I will start with a simple case in one dimension and build up to three dimensions, highlighting the major concepts and terms associated with unit cells.

In One Dimension

Consider the sequence of numbers shown below.

$$\ldots,1,0,1,0,1,0,1,0,1,0,\ldots$$

There are a couple of things that stand out. First, the ellipsis on both ends indicate that the ten numbers we see here are just a small representation of a much larger group that extends in both directions to infinity. Second, there is enough information provided here to create the entire infinite sequence. I can put a copy of these ten numbers right after these ten numbers and then another copy after those ten to get thirty numbers Fig. 1.10.

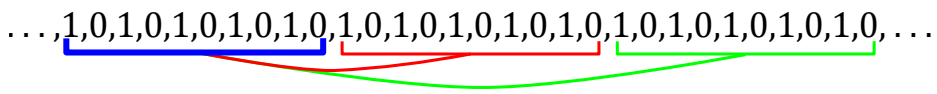


Figure 1.10 Replicating a unit cell with repeat length ten to create a unit cell of repeat length thirty.

By repeating those ten numbers to the left and right, we can generate the entire infinite sequence. A part of the whole that can be used to create the entire whole is called a unit cell. Those ten numbers are a unit cell of repeat length or distance ten; meaning, by repeating those numbers every ten places, we get the whole. In this cases there are an infinite number of unit cells with repeat lengths from two to infinity. Now a primitive unit cell is the smallest unit cell, or the smallest part of the whole that can be used to create the entire whole. In this case there are only two possible primitive unit cells, $[0, 1]$ and $[1, 0]$, both with repeat length two. It does not matter which of the two you use because they both have the same number of elements and are the same size. Notice how the primitive unit cell and repeat length contain all the information needed to construct the entire infinite sequence. We have reduced an infinite sequence to two numbers and a repeat length without loosing any information. It is this ability to reduce the amount of information we need to store while retaining all the information about the whole that makes unit cells so valuable. Another thing to notice, is that other unit cells can be made by

replicating the primitive unit cells an integer number of times. In this case the cells $[0, 1, 0, 1, 0, 1]$ and $[0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]$ can be made by replicating the unit cell $[0, 1]$ three and seven times respectively.

In Two Dimensions

In the case of two dimensions, things get a little more tricky, so we will start with some simple tessellations. Tessellations are large figures that can be generated from smaller parts like the brick patio in Fig. 1.11.

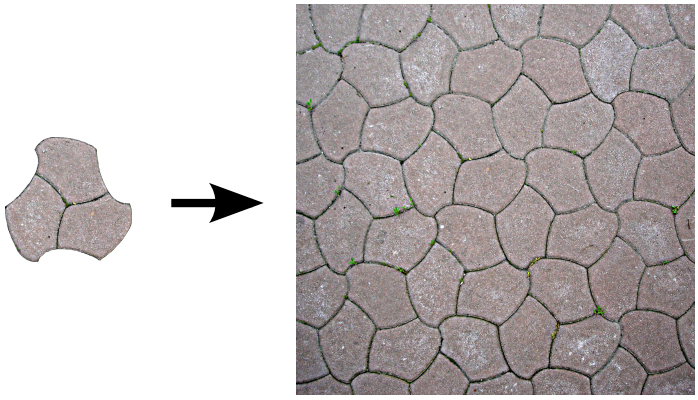


Figure 1.11 Tessellations, like this brick patio, are created by replicating a smaller part, like the three stone unit cell on the left.

When looking at the brick patio above, one might be tempted to use just one stone as the unit cell, but there is a problem with this. If you used a single stone, you would have to rotate your unit cell to produce the whole patio. For it to be a solid state unit cell, the only transformation allowed is translation. Using three stones as a unit cell, the whole patio can be produce by translation only. As with the one dimensional case, each unit cell requires a repeat distance to create the whole, but in two dimensions you need two repeat distances and two repeat directions i.e. two repeat vectors. These repeat vectors are called lattice vectors and are drawn from

one point within a unit cell to an identical point in an adjacent unit cell, as shown in Fig. 1.12.

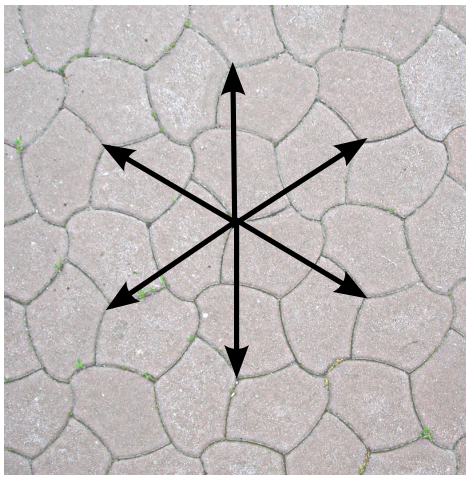


Figure 1.12 Possible lattice vectors for the brick patio

Unlike the one dimensional case, there are more than one possible lattice vector, but which two you use does not matter, because all the other possibilities are linear combinations of any two you choose. Once you have two lattice vectors and a unit cell the whole patio can be generated by placing a copy of the unit cell at all the possible linear combinations of the two lattice vectors. In three dimensions,the magnitude of any linear combination of lattice vectors is a repeat distance, because it is the distance you have to go before the lattice repeats itself. Similar to the one dimensional case, a larger unit cell can be made by replicating the unit cell a finite number of times. However, unlike the one dimensional case, there are numerous possible primitive unit cells. The lattice vectors belonging to primitive unit cells are called primitive lattice vectors. Just three additional primitive unit cells are shown in Fig. 1.13

The last two unit cell examples illustrate an interesting property of unit cells in two dimensions. Any parallelogram shaped region of the whole whose sides are linear combinations of lattice vectors is a unit cell. If the sides are primitive lattice vectors,

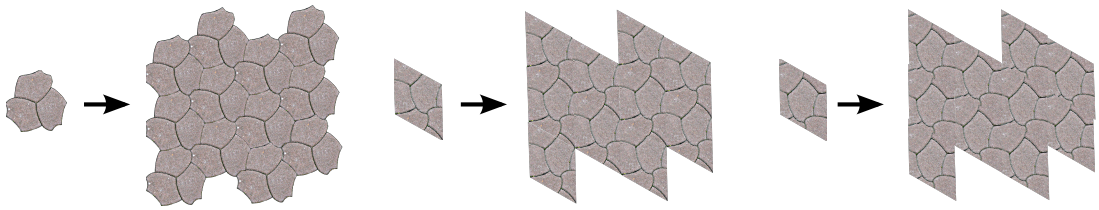


Figure 1.13 Three possible unit cells for the brick patio.

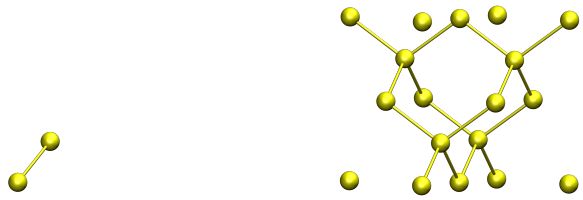
then the unit cell is primitive.

In Three Dimensions

Many of the properties of unit cells in three dimensions can be extrapolated from the one and two dimensional cases. To define how to repeat the unit cell, you need one one-component vector, a scaler, in one dimension, and two two-component vectors in two dimensions. In three dimensions, you need three three-component vectors to define the repeatability of the unit cell. These vectors can be found in the same way they were in the two dimensional case, by drawing vectors from one point in a unit cell to identical points in adjacent unit cells. In this case you need three vectors instead of two, and once again it does not matter which ones you choose; you can get the others by linear combination. Similar to the way any parallelogram regions whose sides are linear combinations of lattice vectors is a unit cell, any parallelepiped region whose edges are linear combinations of of lattice vectors is a unit cell in three dimensions.

Any solid that has a regular repeatable pattern is called a crystal. Crystal unit cells are usually labeled either as primitive or conventional. We have already seen primitive unit cells which are the smallest unit that can be replicated to produce the whole. Conventional unit cells contain a primitive unit cell and a few replications to give a general idea of the shape and relative atomic locations. Usually when showing a

crystals structure you use conventional unit cells to give a better picture of the atomic structure. Figure 1.14 shows a silicon primitive unit cell and a silicon conventional unit cell. If the unit cell is very large it is often called a supercell.



(a) Primitive Unit Cell (b) Conventional Unit Cell

Figure 1.14 Silicon Primitive and Conventional Unit Cells.

Another important concept to understand is fractional or direct atomic coordinates. Often, instead of writing the locations of atoms in the unit cell in Cartesian coordinates, they are written as fractions of the lattice vectors. This makes it easier to see where in the unit cell a certain atom lies from the coordinates and it allows you to use the same coordinates for different elements that share the same crystal structure. All you have to do is change the length of the lattice vectors to account for different bond lengths and the fractional coordinates take care of the relative positions. Going from fractional to Cartesian coordinates is a simple as a matrix transformation Eq. (1.6).

$$\begin{bmatrix} f1_1 & f1_2 & f1_3 \\ f2_1 & f2_2 & f2_3 \\ f3_1 & f3_2 & f3_3 \\ \vdots & \vdots & \vdots \\ fn_1 & fn_2 & fn_3 \end{bmatrix} \times \begin{bmatrix} l1_x & l1_y & l1_z \\ l2_x & l2_y & l2_z \\ l3_x & l3_y & l3_z \end{bmatrix} = \begin{bmatrix} c1_x & c1_y & c1_z \\ c2_x & c2_y & c2_z \\ c3_x & c3_y & c3_z \\ \vdots & \vdots & \vdots \\ cn_x & cn_y & cn_z \end{bmatrix} \tag{1.6}$$

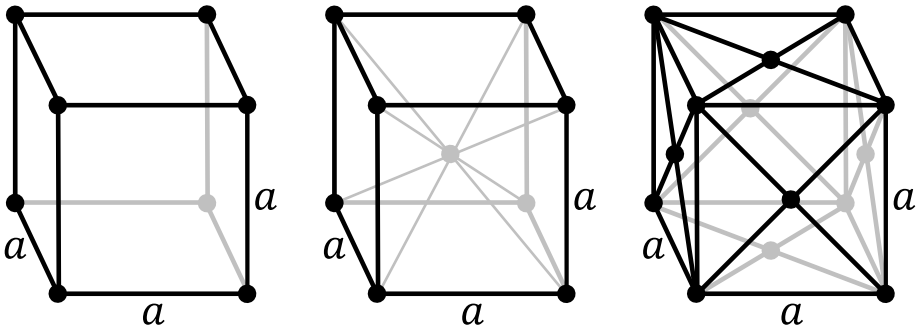
Where the f's are the fractional coordinates, the l's are the Cartesian components of the lattice vectors, and the c's are the Cartesian coordinates.

Another way to think about creating crystals in three dimensions is to start with the lattice vectors. Using the conventional lattice vectors you create a lattice of points, where there is one point at every possible linear combination of the conventional lattice vectors. You then create the crystal by placing a conventional unit cell at each of the points in the lattice. This theoretical lattice is called the Bravais lattice. Crystals are often classified by their Bravais lattices.

Crystal Classification

In three dimensions there are 14 possible Bravais lattices that are divided into 7 categories. I will start with the most simple category, cubic. Cubic lattices have conventional lattice vectors of equal lengths and are orthogonal to each other, i.e. they form a cube. Where in the cube unit cells are placed determines whether the lattice is simple-cubic, body-centered cubic(bcc) or face-centered cubic(fcc) Fig 1.15.

Because they are the simplest, cubic lattices are the Bravais lattices most commonly examined in introductions to crystallography. Silicon is an example of an fcc lattice. If you take the unit cell shown in Fig. 1.14(a) and place it at the points



(a) Simple Cubic (b) Body-centered Cubic (c) Face-centered Cubic

Figure 1.15 The three cubic Bravais lattices, illustrating were unit cells are placed within the cube.

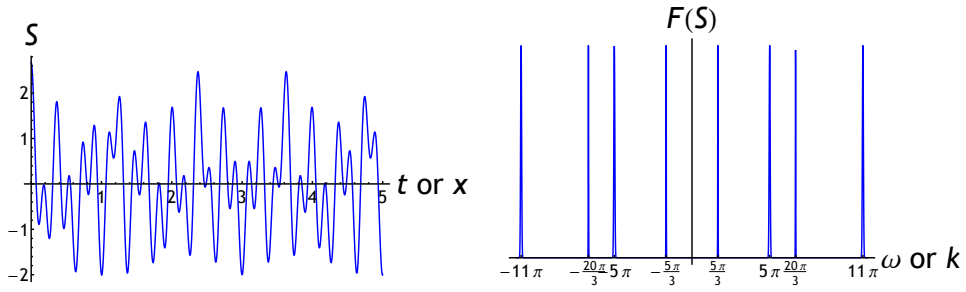
shown in in Fig. 1.15(c) you get something very similar to the conventional unit cell shown in Fig. 1.14(b).

Tetragonal lattices have three orthogonal lattice vectors with two lattice vectors with the same length and one different. Rhombohedral lattices have three lattice vectors with the same length with all three lattice vectors non-orthogonal to each other. Orthorhombic lattices have three orthogonal lattice vectors with three different lengths. Monoclinic lattices have two orthogonal lattice vectors with lattice vectors of any length. Triclinic lattices have three non-orthogonal lattice vectors with lattice vectors of any length. Hexagonal lattices have two lattice vectors of equal length that are 60 degrees apart, and a third lattice vector that is orthogonal to the first two. For graphical representations of all 14 Bravais lattices see Appendix A.

Each Bravais lattice has associated point and space group operations. These are mathematical operations like, reflection, rotation and translations that define the symmetry of the lattice and provide another way of classifying crystal lattices.

1.3.2 Reciprocal Space and The Brillouin Zone

Like unit cells, the Brillouin zone is best understood by extrapolating from lower dimensions. Consider the following signal (S) and its Fourier transform ($\mathcal{F}(S)$) in Fig. 1.16.



(a) Signal (b) Fourier Transform

Figure 1.16 A signal and its Fourier transform.

The Fourier transform in time tells us how much of each frequency(ω) is contained in the signal. Frequency, taken literally tells us how frequent an event, in this case a minimum or maximum occurs. In this particular case there are discrete frequencies that make up our signal because our signal is the sum of cosine waves. This means that minimum and maximum will be one period apart. Alternatively, this signal could be a signal in space and the Fourier transform tells us how much of each spacial frequency(k) is contained in our signal i.e. how far apart local minimum and maximum are ($\lambda = 2\pi/k$). If you were at a maximum, you could expect to find another maximum if you were to look one wavelength to the left or right.

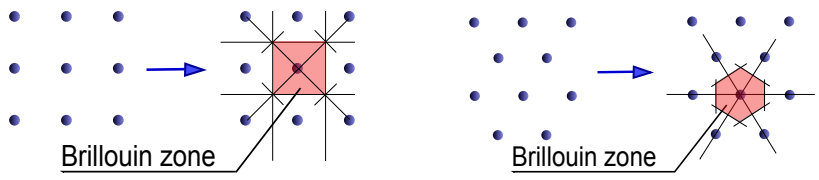
This way of measuring periodicity using Fourier transforms is very helpful in understanding crystal structures. In the same way that we moved from one repeat distance in one dimension to three lattice vectors in three dimensions with the unit cell, we move from a one dimensional Fourier line to a three dimensional Fourier

space. In physics, this space is called the reciprocal space and contains all possible repeat distance and directions. In the same way that periodic signals produce discrete Fourier transforms in one dimension, periodic structures like crystals produce discrete points in reciprocal space. These points form a lattice called the reciprocal lattice and is similar to the Bravais lattice in real space in that there are three reciprocal lattice vectors, and the points in the reciprocal lattice are located at all the possible linear combination of the reciprocal lattice vectors. One elegant aspect of the reciprocal lattice vectors is that they can be calculated from the lattice vectors using Eq. (1.7)

$$\mathbf{b_1} = 2\pi\frac{\mathbf{a_2}\times\mathbf{a_3}}{\mathbf{a_1}\cdot(\mathbf{a2}\times\mathbf{a3})} \quad \mathbf{b_2} = 2\pi\frac{\mathbf{a_3}\times\mathbf{a_1}}{\mathbf{a_1}\cdot(\mathbf{a2}\times\mathbf{a3})} \quad \mathbf{b_3} = 2\pi\frac{\mathbf{a_1}\times\mathbf{a_2}}{\mathbf{a_1}\cdot(\mathbf{a2}\times\mathbf{a3})} \tag{1.7}$$

Now that we have a reciprocal lattice we can define a reciprocal unit cell. There are several ways to define the reciprocal unit cell; I will give just one. Starting with one point in the reciprocal lattice we can draw vectors to all the other reciprocal lattice points. We then place planes normal to these vectors, halfway between the two points. The solid polyhedron bounded by these planes is the reciprocal unit cell. This unit cell is the first Brillouin zone or commonly shortened to the Brillouin zone. The method for constructing the Brillouin zone explained above is illustrated in Fig.1.17 for the case of two dimensions. In short, the Brillouin zone is the unit cell in three-dimensional Fourier(reciprocal) space.

There is one more similarity to the one dimensional Fourier transform. In the one dimensional case, all periodicities can be reduced to corresponding periodicities in any 2π region. The typical regions chosen are $-\pi$ to π and 0 to π. The same thing is true in reciprocal space; any point in reciprocal space can be reduced to a corresponding point in the Brillouin zone. In this way, the Brillouin zone incorporates



(a) Square (b) Hexagonal

Figure 1.17 Constructing the Brillouin zone in two dimensions using lines to neighboring points and normal planes for square and hexagonal reciprocal lattices

the entire reciprocal space.

There is one more reason why the Brillouin zone is so important in solid state physics. In quantum mechanics, each operator has a set of eigenfunctions that can be used as a basis for expressing all possible states for that operator. Also, the de Broglie relations state that momentum is proportional to spacial frequency($p = \hbar k$). The eigenfunctions of the momentum operator are functions of spacial frequency(k), which is exactly what we were looking at in reciprocal space. For that reason, reciprocal space is also called k-space or momentum space, and all possible momentum states can be described as points in k-space. Because all of reciprocal space can be reduced to the Brillouin zone, all possible momentum states can be represented using only points in the Brillouin zone. The eigenfunctions of the momentum operator are also eigenfunctions of the energy operator. Because of this, there is an energy(E) associated with every momentum and hence every point in Brillouin zone. Because of the quantum relationship between momentum and energy, the possible electron states are called bands. The relationship between momentum an energy are shown in band structure plots(see fig. 1.18 for an example), where energy (E) is plotted as a function of spacial frequency (k), which is proportional to momentum.

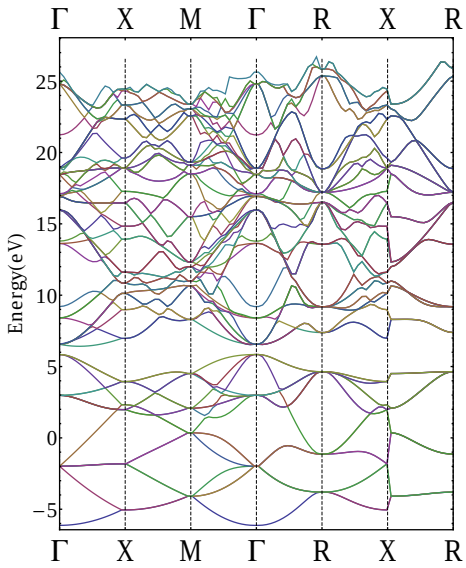


Figure 1.18 Band structure plot for silicon. The symbols on the x-axis represent points in the Brillouin zone. The lines represent the energy levels of states associated with the momentums along the paths from one point in the Brillouin zone to another.

1.3.3 Density of States and the Fermi-level

Density of states is one of the best ways to see the energies of the electrons within a material. As mentioned before, energy and momentum are closely related and we can see the relationship in band plots. One of the best ways to understand density of states is to see how it is related to band structure. Consider the following simplified band structure plot Fig. 1.19.

Instead of plotting solid lines, I have plotted points regularly along the line. Now, we can rotate the plot so that energy is the x-axis Fig. 1.20.

From Fig. 1.20 we can generate a histogram of possible energies Fig. 1.21. This histogram gives a rough idea of how many electrons have a particular energy. One thing to notice is that where the bands are flat, we get more states with similar energies. Additionally, where there are more bands bunched together, there are more

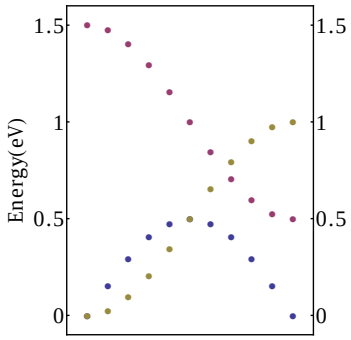


Figure 1.19 A simplified band structure plot with just three bands.

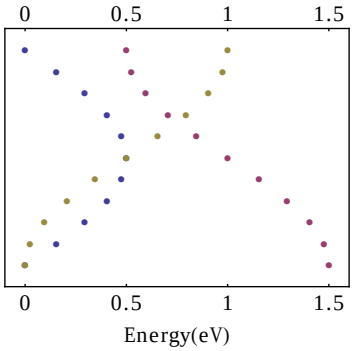


Figure 1.20 A simplified band structure plot rotated so energy is along the x-axis.

states with similar energies.

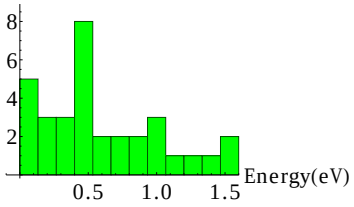


Figure 1.21 A histogram of the band energies in Fig. 1.19.

If we move towards the limit of an infinite number of momentum points in our band plot and infinitesimal energy divisions in our histogram, we get density of states. Density of states tells us how many states of a particular energy level there are. It is typically normalized so that if you integrate the density of states, you get the number

of electrons with energies in region you are integrating over.

Since a material has a finite number of electrons, there are some states that are filled, or have electrons in them, and some that are empty, or do not have electrons in them. In the ground state, or lowest energy configuration, all of the lowest energy states are filled. in the ground state, the highest filled energy is called the Fermi-level.

Chapter 2

Methods

2.1 Identifying Nanotubes to Study

The first step of this project was to identify particular nanotube-silicon systems to study. I began by looking for carbon nanotubes that would be highly commensurate with the silicon lattice. A carbon nanotube is commensurate with the silicon lattice if they have a similar repeat distance ($na_1 \approx ma_2$ see Fig. 2.1). At first, I thought that highly commensurate nanotubes might have special properties that would be interesting to study. Ultimately, this step is necessary to reduce the number of atoms in the unit cell, which would correspond to a dramatic reduction in computational intensity, or the number of processors used times the length of time needed to obtain a certain result. Computational intensity is measured in processor hours and can help quantify the computing power necessary to perform the calculation, and in this study depends on the number of atoms and the volume of the unit cell. For this reason, I was looking for the smallest possible unit cells, which are the unit cells with commensurate nanotube and silicon repeat distances.

I wrote a Matlab program (Appendix B.1) that calculated repeat distances from

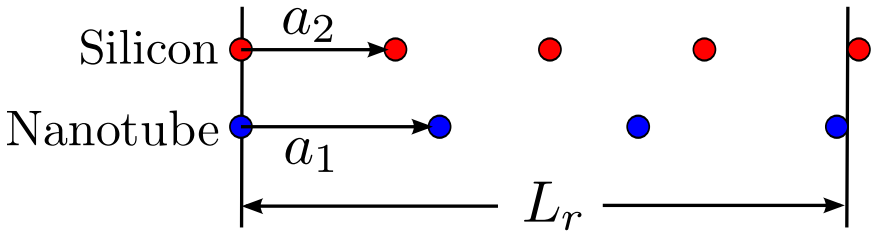


Figure 2.1 A one dimensional representation of a commensurate repeat distance. a_1 and a_2 are nanotube and silicon lattice vectors respectively. L_r is the commensurate repeat distance.

chirality(Eq. (1.2)) and compared multiples of those repeat distances with silicon lattice repeat distances. I obtained the silicon repeat distances by taking the magnitude of many different linear combinations of the three silicon lattice vectors. I quantified the degree of commensurability using strain Eq. (2.1).

$$\text{strain} = \frac{\Delta L_r}{L_r} = \frac{|\text{nanotube repeat distance} - \text{silicon repeat distance}|}{1/2\,(\text{nanotube repeat distance} + \text{silicon repeat distance})} \tag{2.1}$$

I used chirality to calculate the number of atoms for each type of carbon nanotube(Eq. (1.3)), because the computational intensity of the particular calculations I would be doing scale and the number of atoms to the fourth power. My program wrote the strain, number of atoms and other relevant information to an Excel spreadsheet. I selected eight nanotube-silicon systems shown in Fig. 2.2 because they have low strains and relatively small numbers of atoms.

2.2 Creating a Unit Cell

Once I had identified the systems to study, I needed to create a unit cell for each of these systems. This is where commensurability became critical. Carbon nanotubes are essentially one-dimensional structures. While they extend slightly in other direc-

Chirality	% Strain[a]	Multiplier[b]	Num. Atoms[c]	Repeat ($\mathring{A}$)[d]	Direction[e]		
					x	y	z
(3,2)	1.24	1	76	18.81	1	1	1
(4,2)	2.13	1	56	11.52	0	1	1
(6,2)	0.06	1	104	15.36	0	1	1
(7,4)	1.04	1	124	13.85	1	4	5
(7,7)	3.95	3	84	7.68	0	1	1
(7,7)	3.16	4	112	10.16	1	2	3
(8,0)	0.40	1	96	12.74	2	3	3
(8,2)	2.12	1	56	6.65	1	1	2

Figure 2.2 Table of nanotube-silicon systems studied, listed by nanotube chirality. **a.** Strain is the unitless quantity calculated by Eq. (2.1). **b.** Multiplier is the number of nanotube unit cells per system unit cell. **c.** Number Atoms is the number of carbon atoms in the carbon nanotube. **d.** Repeat is the silicon repeat length. The strain can be used to get an idea how close the nanotube repeat length is to this length. **e.** Direction is the commensurate repeat direction in the silicon lattice given in Cartesian coordinates.

tions, they only have one lattice vector.

To study carbon nanotubes embedded in silicon, I needed a unit cell that contains a nanotube and silicon and could be repeated to create nanotubes spread throughout a silicon crystal. To get the silicon crystal, I would have to use the silicon lattice vectors. To get a carbon nanotube, I would have to use the nanotube lattice vector, but the nanotube unit cell can be rotated so the lattice vector points in any direction. If the silicon and nanotube repeat distances are similar, I could use the silicon lattice vector as one of the supercell lattice vectors. This would give a normal silicon crystal containing nanotubes that are strained slightly to match the silicon lattice.

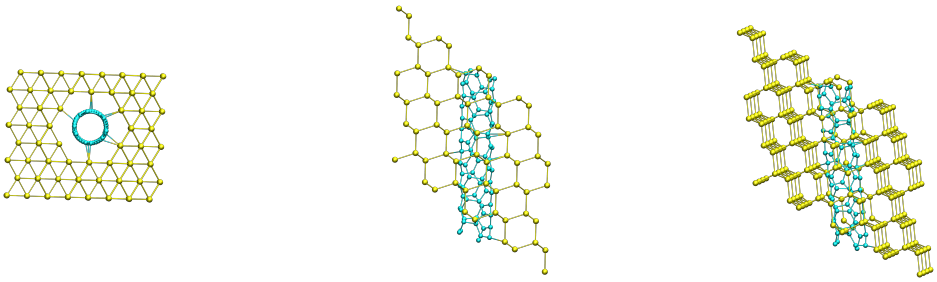
Taking the supercell lattice vector I found using my previous Matlab program, I wrote another program to generate the unit cell (Appendix B.2). The program first selects two additional linear combinations of silicon lattice vectors as supercell lattice

vectors. My code selected supercell lattice vectors that were nearly orthogonal to the first supercell lattice vector vector and to each other. This would give a cube-like unit cell which would allow me to see how the unit cell changed during the calculations and made it easier to include the nanotube. My code also looked for supercell lattice vectors whose xy projection were around ten angstroms in length. In reading through the literature this distance seemed to allow enough separation between neighboring carbon nanotubes to minimize their interaction, allowing me to look at just the interaction between the silicon and the nanotube.

With three supercell lattice vectors, my program generates the coordinates of silicon atoms within a large block of silicon. To help me keep track of the nanotubes lattice vector when performing computations, I chose to rotate the entire block so the commensurate supercell lattice vector points in the z-axis. My program then removes all the silicon atoms within a certain radius of the z-axis to make room for the carbon nanotube. This is where the choice of nearly orthogonal supercell lattice vectors became important. If the two additional supercell lattice vectors are nearly orthogonal, the unit cell has a nearly square cross section when looking down the z-axis . A nearly square cross section gives enough space so the radius chosen to remove silicon atoms does not extend outside of the unit cell. This allowed me to remove atoms only near the z-axis.

The commensurate supercell lattice vector along with the two supercell lattice vectors my program selected, form a parallelepiped. Placing the nanotubes in the center of this parallelepiped, my program then removes all the silicon atoms outside the parallelepiped, leaving me with the silicon part of my unit cell. I used the on-line utility TubeGen [42] to generate the coordinates of the carbon nanotube atoms and combined the nanotube and silicon using Matlab.

To ensure that I had a unit cell, I developed a Matlab program that would replicate the unit cell along the supercell lattice vectors, creating a larger supercell. I then used VMD, a molecular visualization program developed by the Theoretical and Computational Biophysics Group at the University of Illinois at Urbana-Champaign, to visualize the supercell and look for defects. After months of debugging, I had a program that would generate unit cells, like the one shown in Fig. 2.3, and produce the input files needed for performing computations. All eight unit cells can be seen in Fig. 2.4.

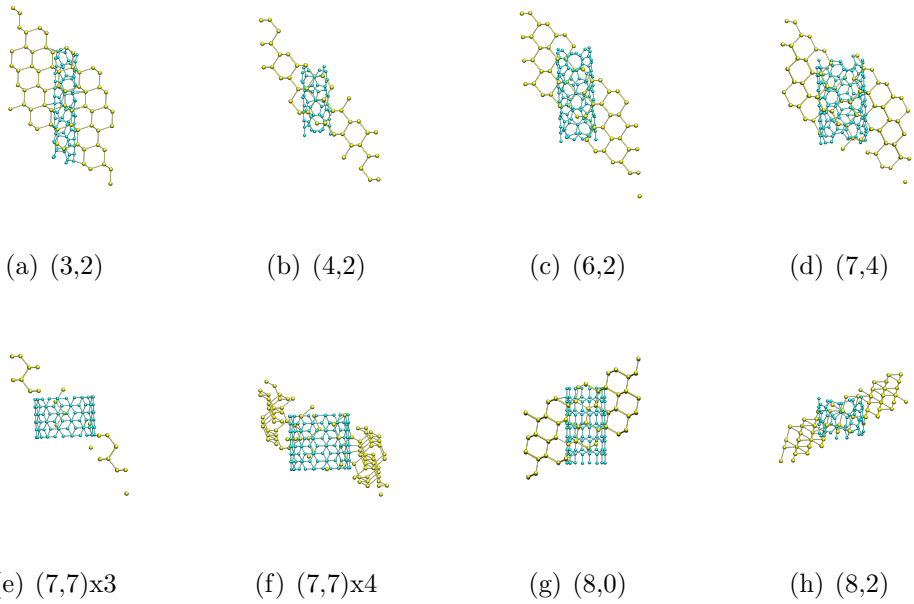


(a) Top View (b) Side View (c) Oblique View

Figure 2.3 Three views of the (3,2) nanotube-silicon unit cell. Note that VMD creates bonds base upon the proximity of atoms and cylinders connecting the nanotube to the silicon may not represent actual bonds.

2.3 Density Functional Theory

Density functional theory is currently the most popular computational method for condensed matter physics, as it greatly reduces the computational requirements of many-atom problems and includes quantum mechanics. It is based on two theorems by Pierre Hohenberg and Walter Kohn that show that: (1) The ground state properties of a many-atom system are uniquely determined by the ground state electron



(a) (3,2) (b) (4,2) (c) (6,2) (d) (7,4)

(e) (7,7)x3 (f) (7,7)x4 (g) (8,0) (h) (8,2)

Figure 2.4 All eight unit cells listed by nanotube chirality. See Fig. 2.2 for additional information.

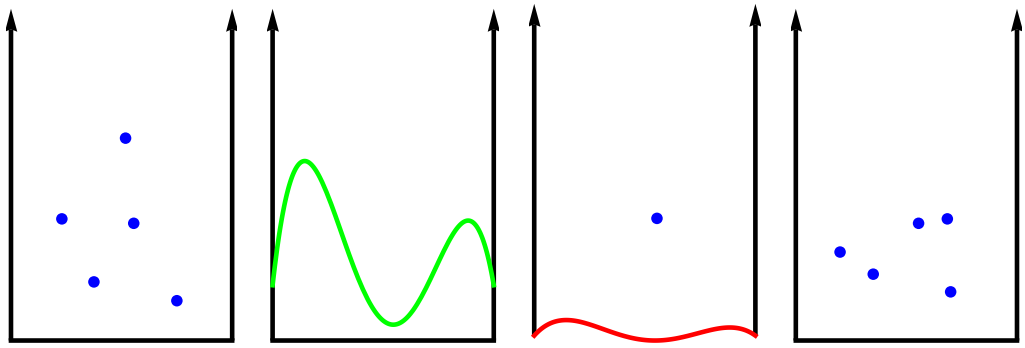
density, and (2) The ground state electron density minimizes an energy functional that can be defined for that system [43]. Walter Kohn later went on to work with Lu Jeu Sham to develop the Kohn-Sham equation. The Kohn-Sham equation reduces a many-electron problem into a problem of single electrons moving in an effective potential that contains the static external potential, and the exchange and correlation interactions [44] These two developments form the basis of density functional theory and the computational methods, like VASP, that implement the theory.

The goal of any computational method based upon density functional theory is to obtain the ground state electron density and use it to calculate the ground state properties of the system. The ground state electron density is obtained by a process that utilizes the variational principle covered in introductory quantum mechanics. You start with an initial guess for the state of all the electrons in your system Fig.2.5(a). Without density functional theory or an approximation, you would have to plug

your initial guess into the Schrödinger equation and reduce the resulting equation to minimize the energy. In most cases this is impractical, because solution to the Schrödinger equation depends on the position of every electron ($\psi(\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}, \ldots, \mathbf{r_n})$). Without density functional theory or an approximation, this is currently impossible for more than a few electrons.

Density functional theory gives an alternative to this intractable differential equation. Using the initial guess we can calculate the electron charge distribution. The charge distribution is combined with the existing potential and exchange and correlation terms to create an effective potential Fig. 2.5(b). The exchange and correlation terms take into account that we are moving from point particles to charge distributions. This allows us to treat each electron independently in the effective potential Fig. 2.5(c). This reduces the original differential equation, to one differential equation for each electron that accounts for the other electrons through the electron charge density. The solutions to these equations depend only on the location of the individual electrons ($\psi_1(\mathbf{r_1})$, $\psi_2(\mathbf{r_2})$, $\psi_3(\mathbf{r_3})$, $\ldots \psi_n(\mathbf{r_n})$). The wave function of each electron in the effective potential is calculated and minimized using the variational principle. This gives a new set of wave functions that can be used as a new guess Fig. 2.5(d). This process is repeated until convergence is reached, or in other words, the change in total energy after one cycle is below a set tolerance. Once convergence is reached the ground state properties can be calculated.

While density functional theory greatly simplifies computation, it does have several limitations. First, the exchange and correlation terms have not been solved exactly for solids, so approximations have to be used. These approximations introduce errors and there is currently no way of knowing how these errors impact the results. Second, density functional theory is a ground state theory and cannot be used to



(a) Initial Guess (b) Electron Density (c) Single Particles (d) New Guess

Figure 2.5 Using density functional theory the ground state electron density can be obtained through the following cyclic process. **a.** Start with an initial guess of the states of the electrons in your system. **b.** Calculate the electron density from your initial guess. **c.** Combine the electron density with the fixed potential and include exchange and correlation effects. Calculate the wave function of each electron in the effective potential independently. **d.** Use the resulting wave functions as a new guess and repeat this process. This cyclic process converges to the ground state electron density.

calculate the excited states of the system. This means that it generally underestimates the band gap of semiconductors. This is most noticeable in calculating band structures. One way to take advantage of density functional theory while obtaining quantitative accuracy is to calculate the band structure using density functional theory and fitting the structure to a few values obtained through a more computationally intensive method.

2.4 VASP

All of my calculations are done on the Fulton Supercomputer at BYU using Viena Ab-initio Package Simulation(VASP). VASP is a first principles quantum-mechanical molecular dynamics package based on density functional theory. Because VASP uses

many matrix operations, it is fairly easy to create a parallel version of VASP that calls the parallel versions of those matrix operations. Two of the features of VASP are that it uses pseudo-potentials and a plane-wave basis.

2.4.1 Pseudopotentials

Pseudopotentials take advantage of the fact that the bonding and interaction between atoms depends mostly on the valence elections. A pseudopotential combines the potential of the nucleus with the potential of electrons in the filled shells. This reduces the number of electrons that VASP considers, reducing computational intensity. There are several methods generating the pseudo-potentials and to ensure that results are consistent and can be related to each other, the VASP group has standardized the pseudopotentials used by VASP.

2.4.2 Plane-wave Basis

In the same way that a musical signal can be compressed into an mp3 through the use of Fourier transforms, the wave function and electron density can be compressed by expressing them in terms of a plane wave basis. The sine and cosine waves($e^{\mathrm{i}\omega t}$) in one dimension are replaced by plane waves($e^{\mathrm{i}\mathbf{k}\cdot\mathbf{r}}$) in three dimensions. VASP expresses both the wave functions and charge density in terms of a plane wave basis, greatly reducing the memory required to run VASP. Unlike with pseudo-potentials, VASP requires that the user specify which plane-waves they want to use for their calculation. Since plane-waves are distinguished by their k-vector, selecting plane waves is identical to selecting points in the Brillouin zone. The points selected are called k-points, because the correspond to a particular k vector. To get a good representation of the entire Brillouin zone, k-points are typically chosen at regular intervals forming a three

dimentional array called the k-mesh Fig. 2.6. In VASP, these points can be explicitly specified or automatically generated. As with sound, if you use more plane-waves you get a more accurate calculation, but one that requires more memory and is more computationally intensive.

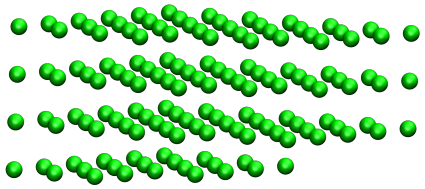


Figure 2.6 These points are an example of an array of k-points in the Brillouin zone that could be chosen for a calculation. The array of k-points chosen is called the k-mesh.

2.5 Two Step Calculations

My goal was to use VASP to obtain the density of states(DOS) for each of the nanotube-silicon systems which I would use to analyze the electronic properties of these systems. Obtaining accurate DOS with VASP requires a two step process. In the first step, the ions are allowed to move until the system is relaxed. VASP begins all runs by using the cyclical method described above to calculate the electronic charge density. Using the charge density, VASP determines the forces on the ions and allows them the move under the influence of those forces. It then calculates the electronic charge density with the ions in a new location and moves the ions again. This process

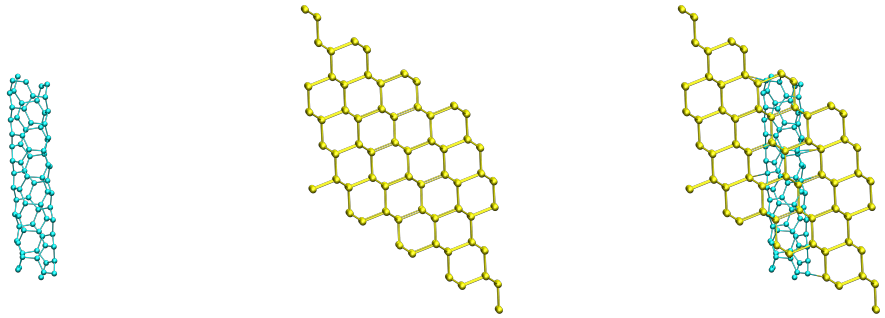
repeats until the change in energy when the ions move is below a specified value. To reduce the computational intensity, this step is usually performed with fewer plane waves from a small number of k-points taken from across the whole Brillouin zone. Usually, this does not affect the accuracy of this step significantly.

The second step is calculating a well-defined electron charge density. During this step the ion locations are fixed and the number of plane waves is increased by using more k-points. Increasing the number of plane waves gives a more accurate DOS. The results of this step can be used to determine the energy of the system and generate density of states plots. There are several settings in VASP that affect the way the calculation is performed and how the DOS looks. I explored these settings using silicon to develop a set of parameters that gave the best density of states. The results of this exploration are included in Appendix C.

If you wanted to calculate the band structure a third step is needed. Rather than including k-points from all of the Brillouin zone, this step requires you to specify a path in Brillouin zone along which VASP calculates the band structure. In this step the charge density is fixed and the k-points are treated independently. Each type of lattice has its own set of high-symmetry paths and reciprocal lattice point labels that are commonly used to determine electronic properties [45] It is important to remember that VASP does not produce quantitatively accurate band structure plots, but the plots it does produce can give general insight into the properties of the crystal.

To calculate the binding energy and see which parts of the combined density of states come from the silicon and which parts come from the nanotube, For each nanotube-silicon system, I generated three subsystems: 1. Carbon nanotube only, 2. Silicon with a hole only and 3. Combined silicon and carbon nanotube system. Fig. 2.7. For each of these subsystems I calculated the total energy and density of states.

The binding energy can be calculated by taking the difference between the sum of the energies when the nanotube and silicon are separate, and the energy when they are together. Calculating the density of states for these three system helped me to see which aspects of the combined density of states come from the nanotube, which come from the silicon, and if there is anything new. VASP has the ability to do site projected density of states, or the density of states belonging to a particular atom, but is limited to only one atom per run. Most of my systems have over two hundred atoms and doing a site projected density of states for every atom would be unfeasible.



(a) Nanotube Only (b) Silicon Only (c) Combined

Figure 2.7 The three different subsystems per nanotube-silicon system.

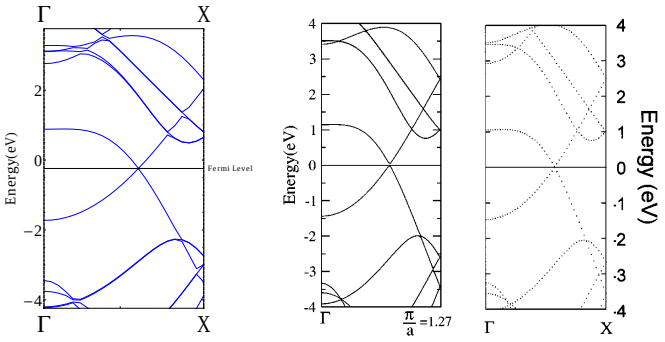
Chapter 3

Results

3.1 Verifying Methods

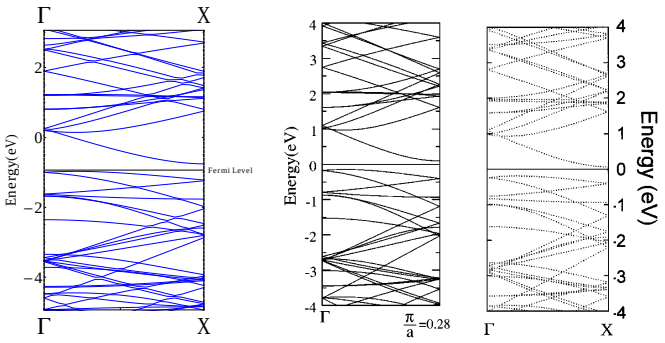
Before I started my computations, I wanted to make sure I knew how to use VASP. I wanted to run some simple jobs involving carbon nanotubes. I looked in the literature, but could not find a published density of states for a single carbon nanotube. I did find band structures for (3,3), (4,2) and (5,0) carbon nanotubes, so I went ahead and used those to verify my method. I relaxed the ion locations, calculated a accurate electron density, and then calculated the band structure. Because carbon nanotubes are essentially one dimensional, I only needed a k-path that ran along the reciprocal vector corresponding to the axis of the tube. I plotted the band structures using the Mathematica code in Appendix B.4. My calculated band structures along with the band structures from two other papers are shown in Fig.3.1-3.3.

I found that my results agree very well with what had been previously published in the literature, assuring me that my methods were correct. There are a few areas on my plots where the band structure seems to jump. This is because VASP stores the bands in order of increasing energy. This means that if two bands cross, they



(a) (3,3) This Work (b) (3,3) [46] (c) (3,3) [47]

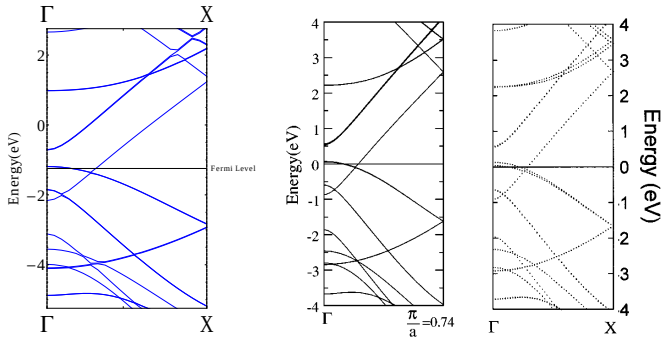
Figure 3.1 Comparing my calculated band structure for (3,3) carbon nanotube to previous research.



(a) (4,2) This Work (b) (4,2) [46] (c) (4,2) [47]

Figure 3.2 Comparing my calculated band structure for (4,2) carbon nanotube to previous research.

switch band numbers. Using the VASP output to plot bands means that a single band line will stay with the same band number, which can correspond to different physical bands.



(a) (5,0) This Work (b) (5,0) [46] (c) (5,0) [47]

Figure 3.3 Comparing my calculated band structure for (5,0) carbon nanotube to previous research.

3.2 Energies

I began my calculations by relaxing the ion locations for each of the twenty-four subsystems. I relaxed the ions until the largest force on any ion was under 0.04 electron-volts per angstrom (eV/Å), which is consistent with what I found in published research [46, 48]. I calculated a high quality electron density for each system and used the total energies for each to calculate the binding energy for each nanotube-silicon system Fig. 3.4.

I thought that the binding energy might be dependent on the orientation of the silicon lattice due to the different number of unbound silicon atoms, or dangling bonds, created by the hole in silicon. The commensurate supercell repeat vectors for the (4,2), (6,2), and (7,7)x3 systems all lie in the (0,1,1) direction, but they have binding energies of 4.50eV, 2.95eV and -2.11eV respectfully. This does not mean that orientation does not affect binding energy, it just indicates that other factors are more significant for these systems.

I also thought that the binding energy might be dependent on the radius of the

System	Nanotube[a]	Silicon[b]	Combined[c]	Binding Energy[d]
(3,2)	-652.57	-1202.15	-1868.53	13.82
(4,2)	-490.85	-720.42	-1215.77	4.50
(6,2)	-934.57	-848.81	-1786.33	2.95
(7,4)	-1127.12	-758.58	-1883.67	-2.04
(7,7)x3	-764.88	-311.68	-1074.45	-2.11
(7,7)x4	-1021.40	-452.01	-1469.53	-3.88
(8,0)	-866.75	-737.43	-1605.38	1.21
(8,2)	-507.74	-434.02	-941.74	-0.02

Figure 3.4 Table of total energies and binding energies for nanotube-silicon systems sorted by nanotube chirality and any multiplier. **a.** Nanotube is the total energy of the relaxed nanotube subsystem. **b.** Silicon is the total energy of the relaxed silicon subsystem. **c.** Combined is the total energy of the nanotube and silicon unit cell. **d.** Binding energy is the negative of the difference between the total energy of the combined system(c) and the sum of the energies of the isolated nanotube(a) and the isolated system(b). A positive number indicates binding.

carbon nanotube. I plotted the binding energy as a function of nanotube radius Fig. 3.5.

The plot shows a strong correlation between binding energy and radius. This is in good agreement with previous studies of single atom adsorbtion on carbon nanotubes, where it was shown that binding energy is inversely related to nanotube radius [49]. The reason for this effect is linked to how the hybridized orbitals of graphene change with curvature. Planar graphene is very inert because the π bonds are well aligned. When graphene is curved, like in a nanotube, the π bonds in one atom are no longer aligned with π bonds in neighboring atoms and are able to hybridize with the orbitals of other atoms. In a nanotube, curvature is inversely proportional to radius. Greater curvatures reduce π bond alignment, allowing higher binding energies.

In looking for a reason for the different binding energies, I looked closely at the

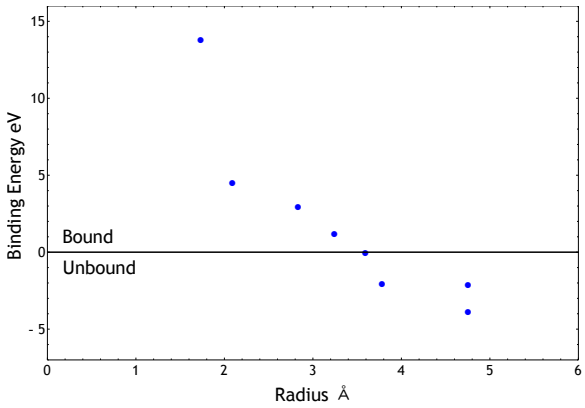


Figure 3.5 Plot of binding energy versus carbon nanotube radius.

relaxed unit cells for the (7,4), (7,7)x3, and (7,7)x4 nanotube-silicon systems. I noticed that the there were only a couple of layers of silicon between neighboring carbon nanotubes. This was the result of how I created my unit cells. The cross-sectional area looking down the z-axis is roughly the same for all my unit cells. For the nanotubes with larger radii, the hole extended nearly to the edge of the unit cell. I need to check if the thinness of silicon between neighboring nanotubes affects binding energy.

3.3 DOS

For the four systems with negative binding energy, I chose not to calculate density of states, because they are harder to create experimentally. For the remaining systems I calculated the density of states using the parameters I found in Appendix C. For this thesis, I have analyzed just the (6,2) nanotube-silicon system. This system had practically no strain (0.06%) and third highest binding energy (2.95eV). The near absence of strain would produce results that show only the interaction between the nanotube and the silicon lattice.

As I looked at the results it became helpful to look at the density of states for just the nanotube and just the silicon parts of the combined unit cell. I took the combined unit cell and extracted the coordinates for the nanotube and calculated the density of states for jointly relaxed nanotube. I also did this for the silicon. This gave me the density of states for the nanotube and silicon with the geometries they have in the combined unit cell. I will refer to the entire nanotube-system as the "combined system", the separate nanotube and silicon systems that were relaxed by themselves as "independently relaxed", and the separate nanotube and silicon systems that were relaxed together and then separated as "jointly relaxed". I could compare these density of states to the density of states for the independently relaxed cases and see how the changes in geometry due to binding affect the density of states. The density of states for the independently and jointly relaxed nanotubes are shown in Figs. 3.6 and 3.7 respectively.

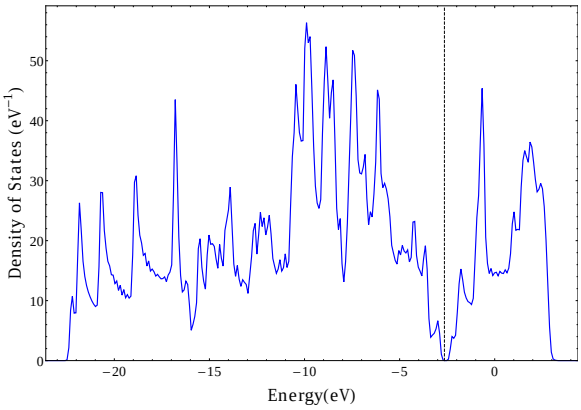


Figure 3.6 DOS plot for independently relaxed (6,2) carbon nanotube. The dashed line represents the Fermi level.

One of the major differences between the two plots is the overall broadening of the peaks in the jointly relaxed nanotube. The broad peaks are likely due to the disorder induced when the atoms in the nanotube are displaced by interactions with

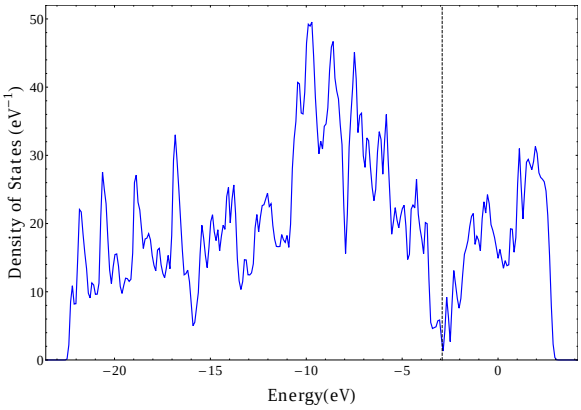


Figure 3.7 DOS plot for jointly relaxed (6,2) carbon nanotube.

the silicon. Another difference is the closing and disappearance of the band gap at the Fermi level. The energy resolution in Fig. 3.7 is 0.093eV, so if the jointly relaxed nanotube has a band gap, it would have to be smaller than that. The disappearance of the gap is also likely caused by disordered created when by relaxing the nanotube.

Before I introduce my results for the silicon subsystems, I want to show the DOS for pure crystalline silicon.

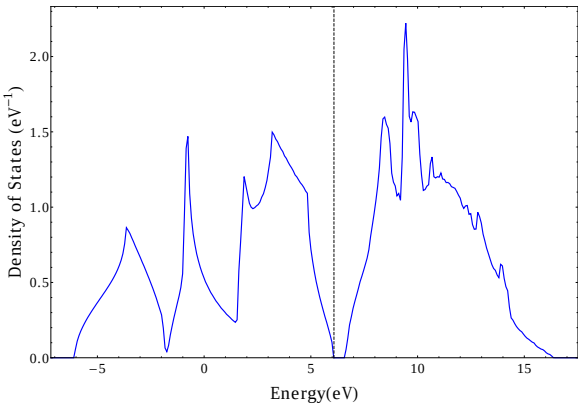


Figure 3.8 Density of states for pure crystalline silicon.

A couple of things to notice are the definite band gap and well defined peaks. Silicon is the model semi-conductor with its 1.11eV band gap. Nearly all semi-conductor devices are made from silicon because it is very inexpensive. The well-defined peaks

are the direct result the well ordered fcc silicon lattice. The density of states is considerably altered by the cutting of a hole to both the independently relaxed Fig. 3.9 and jointly relaxed Fig. 3.10 subsystems.

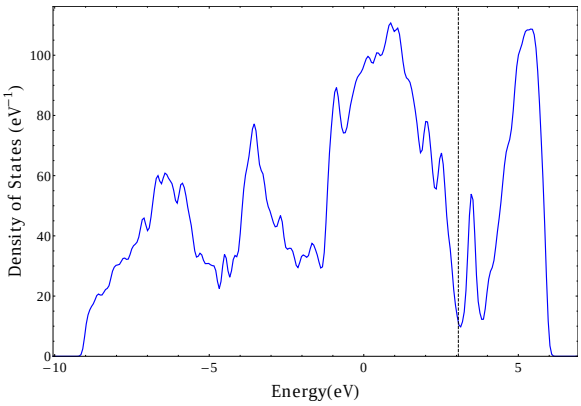


Figure 3.9 Density of states for independently relaxed silicon for the (6,2) nanotube-silicon system.

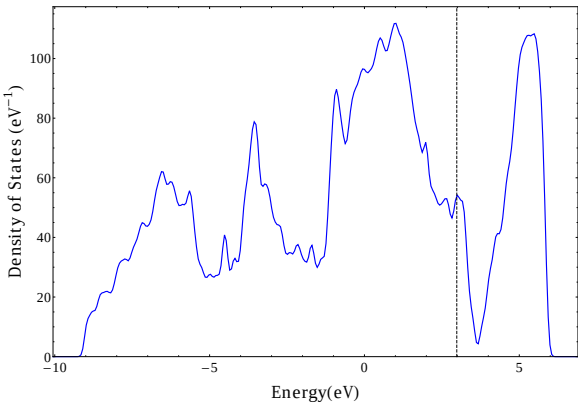


Figure 3.10 Density of states for jointly relaxed silicon for the (6,2) nanotube-silicon system.

In both cases, where the band gap used to be, there are now peaks. These peaks are associated with the unbound silicon atoms on the inside of the hole. These atoms are called dangling bonds, because they are capable of bonding with another atom, but at the moment do not have anything to bond with. These dangling bond states

form a narrow peak, meaning that a group of them have similar energies. In the independently relaxed case, they form their own peak in the gap, above the Fermi-level. In the jointly relaxed case, they form a peak that blends with the peak just lower in energy, right at the Fermi-level. The presence of the nanotube causes structural changes that move the states associated with the dangling bonds to lower energy levels.

The density of states for the combined nanotube-silicon system Fig. 3.11, looked a lot like the addition of the separated nanotube and silicon density of states.

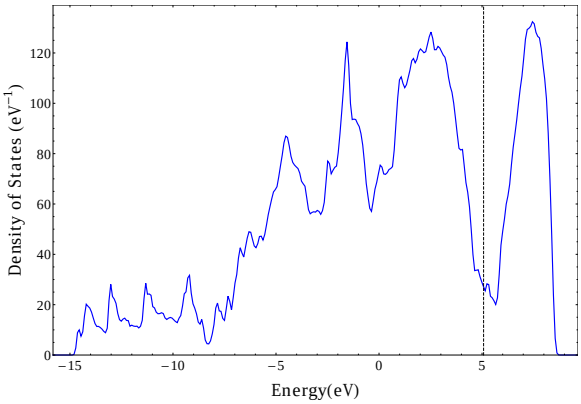


Figure 3.11 Density of states for (6,2) combined nanotube-silicon system.

I wanted to see how much of the combined density of states was just the addition of the nanotube and silicon states. I added the density of states for the independently relaxed nanotube to the density of states for the independently relaxed silicon. I did the same for the jointly relaxed subsystems and plotted the two added density of states along with the calculated density of states for the combined system as shown in Fig. 3.12

I had to shift both the nanotube and the silicon so the peaks would match up with the peaks in the combined system. I used the smaller, lower energy peaks to fit the nanotube and the larger, higher energy peaks to fit the silicon. The shift is

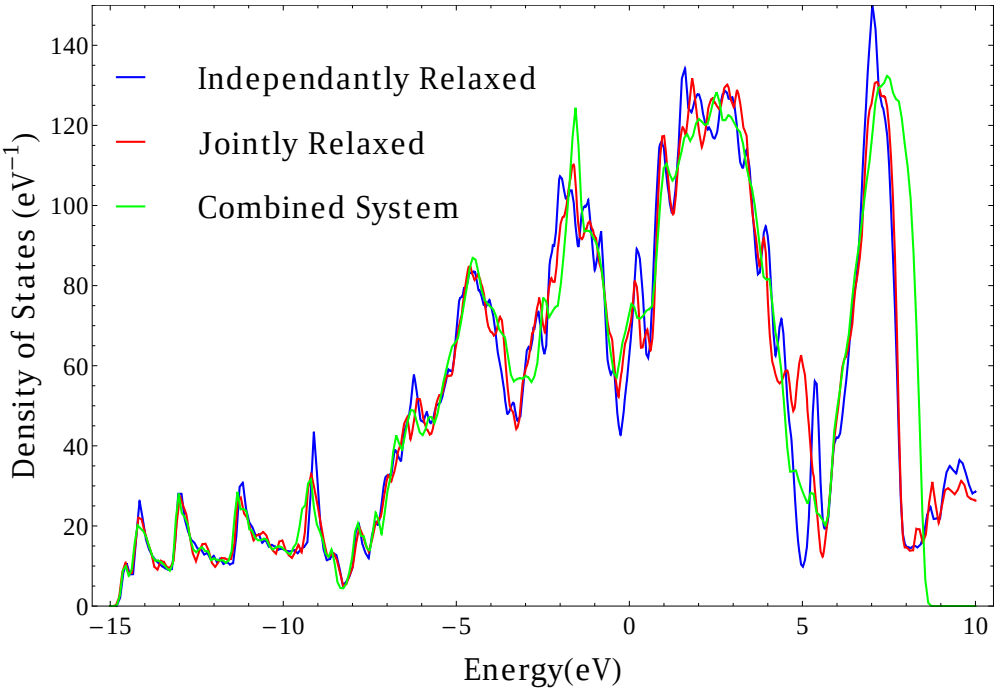


Figure 3.12 Density of states for independently relaxed silicon for the (6,2) nanotube-silicon system.

necessary due to some ambiguity in how VASP defines the zero in energy; however, the shift does appear to be related to the difference in Fermi-levels Fig. 3.13.

Shifting the density of states for the subsystems by about the difference in the Fermi levels makes sense if you think about what happens when you bring two materials with different Fermi levels together. Electrons from the material with the higher Fermi level will flow into lower energy, unoccupied states in the other material, raising its Fermi level and lowering the Fermi level of the donor material. Electrons will continue to flow until the Fermi level of both materials are equal. The Fermi level of the whole system will be somewhere between the two original Fermi levels. In the case of Fig. 3.13, the numbers do not show this because of systematic changes in the zero of energy, but qualitatively we can see this happening. Because the shifts are so close to the differences in the Fermi-level, Fig. 3.12 is roughly aligning the

Subsystem	Fermi Level[a]	Difference[b]	Shift[c]
Combined System	5.07	0	0
Independently Relaxed Nanotube	-2.64	7.72	7.68
Independently Relaxed Silicon	3.06	2.02	1.88
Jointly Relaxed Nanotube	-2.91	7.99	7.64
Jointly Relaxed Silicon	2.98	2.09	1.91

Figure 3.13 Table of Fermi levels, differences, and shifts used to fit the densities of states in Fig. 3.12 showing the relationship between the difference in Fermi level and the shift. **a.** Fermi level is the calculated Fermi level for that subsystem. **b.** Difference is the difference between the Fermi level of the subsystem and the Fermi level of the combined nanotube-silicon system. **c.** Shift is how far I had to shift the density of states to get the best fit in Fig. 3.12.

Fermi levels of the two density of states and adding them together. This suggests that very few electrons flowed between the nanotube and silicon, because, if a large number of electrons flowed then the Fermi level of both the nanotube and the silicon would have changed dramatically. Because we can pretty much place the Fermi levels of both the nanotube and the silicon at the Fermi level of the combined system and have the density of states line up, we know that the Fermi level of both the silicon and nanotube subsystems did not change very much when they were placed together. Since the Fermi level did not change very much, very few electrons flowed between the nanotube and silicon.

Looking at Fig. 3.12, the combined system seems pretty much like the addition of the nanotube and silicon except near the Fermi level. The peaks near the Fermi level that are present in both silicon systems seemed to be smoothed out in the combined system. As I mentioned before, the states in the gap are associated with dangling bonds. The fact that the dangling bond states have smoothed out suggests that the dangling bond states have become delocalized. Our hypothesis is that the different

dangling bond states are hybridizing through the nanotube, creating new band-like states. This means that electrons from one dangling bond can now move to another dangling bond through the nanotube.

Chapter 4

Conclusion

Carbon nanotubes are one of the most exciting materials of this century, possessing singular mechanical, electrical and thermal properties. Before carbon nanotubes can make their way into practical applications, we need a greater understanding of their interactions with other materials. This study shows that the energetics of nanotube-silicon interaction is highly dependent upon nanotube curvature. In addition, this study provides evidence that very few electrons are shared between the nanotube and silicon and suggests that the nanotube facilitates silicon dangling bond hybridization. Silicon is currently the most common semiconductor. Understanding how it interacts with carbon nanotubes is the first step in producing practical nanotube-silicon devices. Unlike previous studies that examine the interaction between carbon nanotubes and single atoms, this study examines bulk carbon nanotube materials extending theoretical predictions to a scale that can be more easily observed in experiments. This study narrows the current gap between theoretical and experimental research.

Future work on this project would include analyzing the DOS for the (3,2), (4,2), and (8,0) systems to see if low electron flow and dangling bond hybridization are common to all nanotube-silicon systems, or if each has its own unique behavior.

Additionally, I would like to see if increasing the amount of silicon between carbon nanotubes yields positive binding energies for the (7,4), (7,7)x3, (7,7)x4, and (8,2) cases. Future work could include calculating the band structure for each system, which would give greater understanding into how the silicon band gap changes with the addition of carbon nanotubes. Using the tools I have already developed, it would be fairly easy to preform the same calculations for doped silicon and other semiconductors like galium nitride. Alternatively, other nanotubes like gallium phosphide could be used instead of carbon nanotubes. As greater computing resources become available, it will be possible to look at larger nanotubes. Ultimately, it would be possible to use the methods developed during this project to understand the electronic properties of any nanotube embedded in any crystal.

Bibliography

[1] X. Wang, Q. Li, J. Xie, Z. Jin, J. Wang, Y. Li, K. Jiang, and S. Fan, "Fabrication of Ultralong and Electrically Uniform Single-Walled Carbon Nanotubes on Clean Substrates," Nano Letters **9,** 3137–3141 (2009).

[2] S. Iijima, "Helical microtubules of graphitic carbon," Nature **354,** 56–58 (1991).

[3] T. Guo, P. Nikolaev, A. Thess, D. Colbert, and R. Smalley, "Catalytic growth of single-walled manotubes by laser vaporization," Chemical Physics Letters **243,** 49–54 (1995).

[4] M. Jose-Yacaman, M. Miki-Yoshida, L. Rendon, and J. G. Santiesteban, "Catalytic growth of carbon microtubules with fullerene structure," Appl. Phys. Lett. **62,** 657–659 (1993).

[5] M. Monthioux and V. L. Kuznetsov, "Who should be given the credit for the discovery of carbon nanotubes?," Carbon **44,** 1621–1623 (2006).

[6] J. W. Mintmire, B. I. Dunlap, and C. T. White, "Are fullerene tubules metallic?," Phys. Rev. Lett. **68,** 631 (1992).

[7] W. Marx and A. Barth, "Carbon nanotubes A scientometric study," phys. stat. sol. (b) **245,** 2347–2351 (2008).

[8] A. H. Nevidomskyy, G. Csányi, and M. C. Payne, "Chemically Active Substitutional Nitrogen Impurity in Carbon Nanotubes," Phys. Rev. Lett. **91,** 105502 (2003).

[9] K. K. S. Lau, J. Bico, K. B. K. Teo, M. Chhowalla, G. A. J. Amaratunga, W. I. Milne, G. H. McKinley, and K. K. Gleason, "Superhydrophobic Carbon Nanotube Forests," Nano Letters **3,** 1701–1705 (2003).

[10] M. Shokrieh and R. Rafiee, "A review of the mechanical properties of isolated carbon nanotubes and carbon nanotube composites," Mechanics of Composite Materials **46,** 155–172 (2010).

[11] Y. H. Yang and W. Z. Li, "Radial elasticity of single-walled carbon nanotube measured by atomic force microscopy," Applied Physics Letters **98,** 041901 (2011).

[12] R. S. Ruoff, J. Tersoff, D. C. Lorents, S. Subramoney, and B. Chan, "Radial deformation of carbon nanotubes by van der Waals forces," Nature **364,** 514–516 (1993).

[13] S. Ogata and Y. Shibutani, "Ideal tensile strength and band gap of single-walled carbon nanotubes," Phys. Rev. B **68,** 165409 (2003).

[14] M.-F. Yu, B. S. Files, S. Arepalli, and R. S. Ruoff, "Tensile Loading of Ropes of Single Wall Carbon Nanotubes and their Mechanical Properties," Phys. Rev. Lett. **84,** 5552 (2000).

[15] M.-F. Yu, O. Lourie, M. J. Dyer, K. Moloni, T. F. Kelly, and R. S. Ruoff, "Strength and Breaking Mechanism of Multiwalled Carbon Nanotubes Under Tensile Load," Science **287,** 637–640 (2000).

[16] K. Jensen, W. Mickelson, A. Kis, and A. Zettl, "Buckling and kinking force measurements on individual multiwalled carbon nanotubes," Phys. Rev. B **76,** 195436 (2007).

[17] T. W. Tombler, C. Zhou, L. Alexseyev, J. Kong, H. Dai, L. Liu, C. Jayanthi, M. Tang, and S.-Y. Wu, "Reversible electromechanical characteristics of carbon nanotubes under local-probe manipulation," Nature **405,** 769 (2000).

[18] C. D. Spataru and F. Léonard, "Tunable Band Gaps and Excitons in Doped Semiconducting Carbon Nanotubes Made Possible by Acoustic Plasmons," Phys. Rev. Lett. **104,** 177402 (2010).

[19] L. Chico, V. H. Crespi, L. X. Benedict, S. G. Louie, and M. L. Cohen, "Pure Carbon Nanoscale Devices: Nanotube Heterojunctions," Phys. Rev. Lett. **76,** 971 (1996).

[20] H. W. C. Postma, T. Teepen, Z. Yao, M. Grifoni, and C. Dekker, "Carbon Nanotube Single-Electron Transistors at Room Temperature," Science **293,** 76–79 (2001).

[21] H. J. Dai, A. Javey, E. Pop, D. Mann, W. Kim, and Y. R. Lu, "Electrical transport properties and field effect transistors of carbon nanotubes," Nano **1,** 1–13 (2006).

[22] T. Durkop, S. A. Getty, E. Cobas, and M. S. Fuhrer, "Extraordinary mobility in semiconducting carbon nanotubes," Nano Letters **4,** 35–39 (2004).

[23] S. Berber, Y. K. Kwon, and D. Tomanek, "Unusually high thermal conductivity of carbon nanotubes," Physical Review Letters **84,** 4613–4616 (2000).

[24] E. Pop, D. Mann, Q. Wang, K. Goodson, and H. J. Dai, "Thermal conductance of an individual single-wall carbon nanotube above room temperature," Nano Letters **6,** 96–100 (2006).

[25] S. Sinha, S. Barjami, G. Iannacchione, A. Schwab, and G. Muench, "Off-axis thermal properties of carbon nanotube films," Journal of Nanoparticle Research **7,** 651–657 (2005).

[26] C. Suplee, "Infintesimal Carbon Structures May Hold Gigantic Potential," Washington Post **June 2,** A3 (1996).

[27] L. Yeung, "A giant advance, just atoms wide; Nanotechnology is a scientific holy grail for the 21st century - and HKUST is pioneering its research," South China Morning Post (Hong Kong) **July 7,** 3 (2001).

[28] M. Endo, M. S. Strano, and P. M. Ajayan, "Potential applications of carbon nanotubes," Carbon Nanotubes **111,** 13–61 (2008).

[29] K. W. C. Lai, N. Xi, C. K. M. Fung, J. B. Zhang, H. Z. Chen, Y. L. Luo, and U. C. Wejinya, "Automated Nanomanufacturing System to Assemble Carbon Nanotube Based Devices," International Journal of Robotics Research **28,** 523–536 (2009).

[30] M. S. Arnold, A. A. Green, J. F. Hulvat, S. I. Stupp, and M. C. Hersam, "Sorting carbon nanotubes by electronic structure using density differentiation," Nature Nanotechnology **1,** 60–65 (2006).

[31] T. Tanaka, Y. Urabe, D. Nishide, and H. Kataura, "Continuous Separation of Metallic and Semiconducting Carbon Nanotubes Using Agarose Gel," Applied Physics Express **2,** 125002 (2009).

[32] X. Y. Huang, R. S. McLean, and M. Zheng, "High-resolution length sorting and purification of DNA-wrapped carbon nanotubes by size-exclusion chromatography," Analytical Chemistry **77,** 6225–6228 (2005).

[33] X. M. Tu, S. Manohar, A. Jagota, and M. Zheng, "DNA sequence motifs for structure-specific recognition and separation of carbon nanotubes," Nature **460,** 250–253 (2009).

[34] A. K. Geim, S. V. Dubonos, I. V. Grigorieva, K. S. Novoselov, A. A. Zhukov, and S. Y. Shapoval, "Microfabricated adhesive mimicking gecko foot-hair," Nature Materials **2,** 461–463 (2003).

[35] L. Ge, S. Sethi, L. Ci, P. M. Ajayan, and A. Dhinojwala, "Carbon nanotube-based synthetic gecko tapes," Proceedings of the National Academy of Sciences of the United States of America **104,** 10792–10795 (2007).

[36] B. Yurdumakan, N. R. Raravikar, P. M. Ajayan, and A. Dhinojwala, "Synthetic gecko foot-hairs from multiwalled carbon nanotubes," Chemical Communications **2005,** 3799–3801 (2005).

[37] Y. Zhao, T. Tong, L. Delzeit, A. Kashani, M. Meyyappan, and A. Majumdar, "Interfacial energy and strength of multiwalled-carbon-nanotube-based dry adhesive," Journal of Vacuum Science & Technology B **24,** 331–335 (2006).

[38] S. Sethi, L. Ge, L. Ci, P. M. Ajayan, and A. Dhinojwala, "Gecko-inspired carbon nanotube-based self-cleaning adhesives," Nano Letters **8,** 822–825 (2008).

[39] L. Xiao *et al.*, "Flexible, Stretchable, Transparent Carbon Nanotube Thin Film Loudspeakers," Nano Letters **8,** 4539–4545 (2008).

[40] V. N. Popov, V. E. Van Doren, and M. Balkanski, "Elastic properties of single-walled carbon nanotubes," Physical Review B **61,** 3078–3084 (2000).

[41] D. N. Hutchison, N. B. Morrill, Q. Aten, B. W. Turner, B. D. Jensen, L. L. Howell, R. R. Vanfleet, and R. C. Davis, "Carbon Nanotubes as a Framework for High-Aspect-Ratio MEMS Fabrication," Journal of Microelectromechanical Systems **19,** 75–82 (2010).

[42] J. T. Frey and D. J. Doren, "TubeGen 3.3,", Web, 2005.

[43] P. Hohenberg and W. Kohn, "Inhomogeneous Electron Gas," Phys. Rev. **136,** B864 (1964).

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

[45] W. Setyawan and S. Curtarolo, "High-throughput electronic band structure calculations: Challenges and tools," Computational Materials Science **49,** 299–312 (2010).

[46] M. Machon, S. Reich, C. Thomsen, D. Sanchez-Portal, and P. Ordejon, "Ab initio calculations of the optical properties of 4-angstrom-diameter single-walled nanotubes," Physical Review B **66,** 155410 (2002).

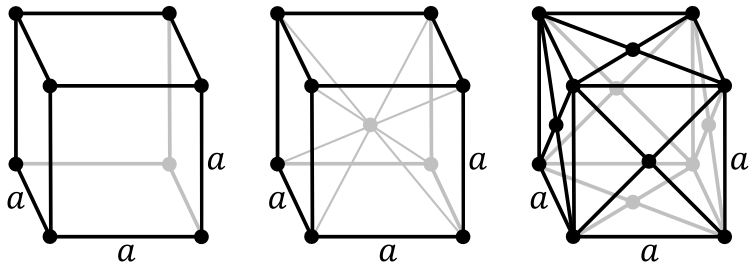
[47] Z. M. Li *et al.*, "Polarized absorption spectra of single-walled 4 angstrom carbon nanotubes aligned in channels of an AlPO4-5 single crystal," Physical Review Letters **87,** 127401 (2001).

[48] A. Ayuela, L. Chico, and W. Jaskólski, "Electronic band structure of carbon nanotube superlattices from first-principles calculations," Phys. Rev. B **77,** 085435 (2008).

[49] O. Gülseren, T. Yildirim, and S. Ciraci, "Tunable Adsorption on Carbon Nanotubes," Phys. Rev. Lett. **87,** 116802 (2001).

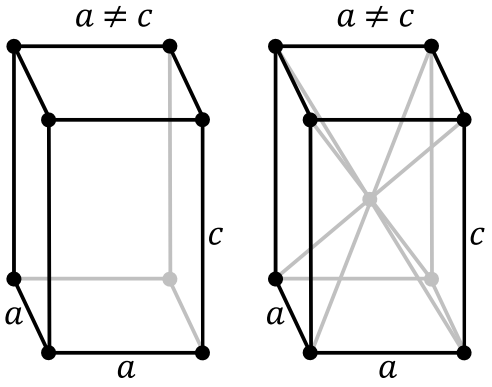
Appendix A

Graphical Representations of Bravais Lattices



(a) Simple (b) Body-centered (c) Face-centered

Figure A.1 Cubic



(a) Simple (b) Body-centered

Figure A.2 Tetragonal

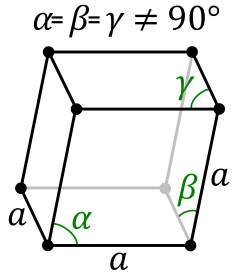
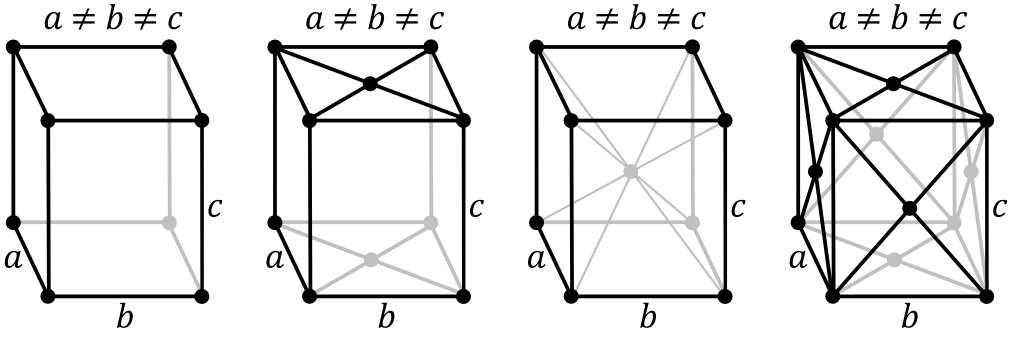
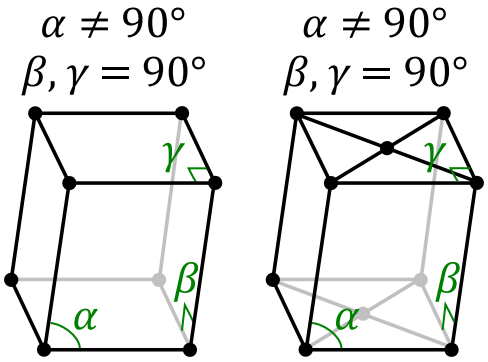


Figure A.3 Rhombohedral



(a) Simple (b) Base-centered (c) Body-centered (d) Face-centered

Figure A.4 Orthorhombic



(a) Simple (b) Base-centered

Figure A.5 Monoclinic

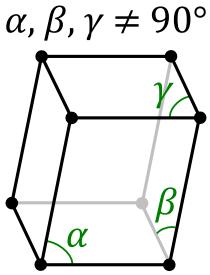


Figure A.6 Triclinic

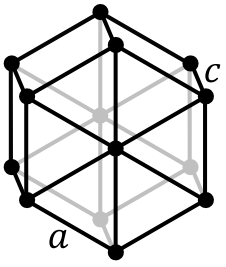


Figure A.7 Hexagonal

Appendix B

Matlab and Mathematica Programs

B.1 CommensurateFinder.m

```
% CommensurateFinder is a Matlab m-file written to compare silicon
% repeat distances with carbon nanotube repeat distances and generate
% a table sorted by the strain =100*abs(p*a-q*srd)/((p*a+q*srd)/2)
% where:
% p and q = interger multipliers
% a = the nanotube repeat distance
% srd = the silicon repeat distance
% Basic Process:
% Start with a carbon nanotube and compare its repeat distance to many
% possible silicon repeat distances. If the strain is below a certain
% value, record relavant information. Repeat this comparing, checking
% strain, and recording relavant information for many nanotubes. Take
% the recorded information and eliminate any duplicates. Write the
% data to file.

clear; close all; clc;

%Set the lattice constants for carbon and silicon
a0C=2.461;   %carbon
a0SC=5.4307; %silicon

%Define the silicon latice vectors
```

```
a1=[a0SC/2,a0SC/2,0];
a2=[a0SC/2,0,a0SC/2];
a3=[0,a0SC/2,a0SC/2];

%Fill a matrix with the latice vectors: row 1 for a1, row 2 for a2,
%row 3 for a3
SRD(1,:)=a1;
SRD(2,:)=a2;
SRD(3,:)=a3;

%Initialize the vector cordinates
xc=0; yc=0; zc=0;

%Set the maximum value for p and q
z=6;

%Assign an index value for the iterations
index=1;

% The following nested FOR loops cycle through various nanotubes and
% silicon repeat directions. The iterators n1 and n2 correspond to the
% chirality of the nanotube(n1,n2). The iterators l1, l2, and l3
% correspond to components of the silicon latice vectors. The silicon
% repeat vector is l1*a1+l2*a2+l3*a3. The iterators p and q correspond
% to interger multipliers of the nanotube and silicon repeat distance.

for n1=3:10 %cycle through n1 values
 for n2=0:10 %cycle through n2 values

     %calculate some useful values related to carbon nanotubes(N,n,R)
     N=n1^2+n1*n2+n2^2;
     n=gcd(n1,n2);
     if mod((n1-n2)/(3*n),1)==0
         R=3;
     else
         R=1;
     end

     %calculate the nanotube repeat distance
     a=sqrt(3*N)*a0C/(n*R);

  %cycle through linear combinations of the silicon lattice vectors
  for l1=0:5
   for l2=0:5
    for l3=0:5

        % Calculate the x,y,z componants of the silicon repeat vector
        xc=l1*SRD(1,1)+ l2*SRD(2,1)+l3*SRD(3,1);
        yc=l1*SRD(1,2)+ l2*SRD(2,2)+l3*SRD(3,2);
        zc=l1*SRD(1,3)+ l2*SRD(2,3)+l3*SRD(3,3);
```

```
        % Calculate the length of the silicon repeat vector
        srd=sqrt(xc^2+yc^2+zc^2);

         for q=1:z %cycle through nanotube multipliers
          for p=1:z %cycle through lattice multipliers

           % Calculate the strain
           strain =100*abs(p*a-q*srd)/((p*a+q*srd)/2);

           % Calculate the number of carbon atoms per nanotube unit
           % cell
           nc=4*N/(n*R);

           % Check if the strain is below a certain percent
            if strain<5 && a*p<50 && nc*p<150

             % Fill a matrix with data about the particular nanotube
             % and silicon repeat vector
             S(index,1)=strain;          % Strain
             S(index,2)=q*srd;           % Silicon Repeat Length
             S(index,3)=nc*p;            % Number of carbon atoms per
                                         %  nanotube unit cell
             S(index,4)=n1;              % n1
             S(index,5)=n2;              % n2
             S(index,6)=p;               % nanotube multiplier
             S(index,7)=q;               % silicon multiplier
             S(index,8)=xc/srd;          % x repeat direction
             S(index,9)=yc/srd;          % y repeat direction
             S(index,10)=zc/srd;         % z repeat direction
             S(index,11)=l1;             % silicon componant l1
             S(index,12)=l2;             % silicon componant l2
             S(index,13)=l3;             % silicon componant l3
             S(index,14)=srd;            % unmultiplied silicon repeat
                                         % distance
             S(index,15)=a;              % unmultiplied nanotube repeat
                                         % distance
             index=index+1;
            end
           end
          end
         end
        end
       end
      end
    end
end

% The following section uses three different tests to eliminate any
% duplicates generated due to symmetry. One cause of duplicates is
% permutations of the same values(i.e.(3,2)=(2,3)).The first two tests
```

```
% cover this case. The other case is a double repeat distance with
% half the multiplier. The third test covers this case.

% Add the possible permuted values
S(:,17)=S(:,2)+S(:,4)+S(:,5)+S(:,8)+S(:,9)+S(:,10)+10000;

% Multiply the possible permuted values plus one. Adding one takes in
% to account the possibility of a zero value, in which case the whole
% expression would be zero.
S(:,18)=(S(:,2)+1).*(S(:,4)+1).*(S(:,5)+1).*(S(:,8)+1).*(S(:,9)+1)...
    .*(S(:,10)+1)+10000;

% Divide the repeat distance and number of atoms per nanotube unit
% cell by the relavant multiplier.
S(:,19)=S(:,3)./S(:,6)+S(:,2)./S(:,6)+S(:,4)+S(:,5)+S(:,8)+S(:,9)+...
    S(:,10)+10000;

% The nested WHILE loops below are designed to keep track of the
% currentlength of the matrix while comparing the entries and
% eliminates any that entries that have the same values for the first
% two tests mentioned above. It starts with the first entry and
% compares it to all the lower entries, eliminating any duplicates. It
% then takes the second entry and compares it to all the lower entries
% eliminating any duplicates. It continues with subsequent entries
% until it reaches the end. Since the entries are  generated in order
% of increasing multipliers, the first entries in the matrix will be
% the simplest. The iterator i corresponds to the entry that is being
% compared to the others. The iterator n corresponds to the entry
%that is being compared to the entry corresponding to i.
l=length(S); % Store the current length of the matrix that will be
             % used to keep track of the current length of the matrix
p=length(S); % Store an additional copy of the length
i=1; % initialize an iterator

%Step through the entries to compare
while i<=l
    n=i+1;
    while n<=p
        %Compare the entries
        if S(i,17)==S(n,17) && S(i,18)==S(n,18)
            % If they are the same, delete the the entry, adjust the
            % current length of the matrix and the entry that is being
            % compared.
            S(n,:)=[];
            p=p-1; % Current length
            l=l-1; % Current length
            n=n-1; % Entry that is being compared. Since an entry was
                   % deleted the next entry to be compared has the
                   % same index and the entry that was deleted. This
                   % is taken into account by moving the iterator back
```

```
                    % an entry.
        end
        % Advance to the next entry
        n=n+1;
    end
    % Advance to the next entry
    i=i+1;
end

% Since the third test checks for a different cause of duplication
% another loop is required. This loop follows the same sturcture as
% the loop above except it compares the value for the third test.
l=length(S);
p=length(S);
i=1;
while i<=l
    n=i+1;
    while n<=p
        if S(i,19)==S(n,19)
            S(n,:)=[];
            p=p-1;
            l=l-1;
            n=n-1;
        end
        n=n+1;
    end
    i=i+1;
end

%Sort the data by the strain
Matrix=sortrows(S,1);

%Generate a list of column labels
label={'Strain','Repeat Length','Number of Atoms','n1','n2',...
    'Nano Repeat Factor', 'Si Repeat Factor','x-repeat',...
    'y-repeat','z-repeat','a1 multiplier','a2 multiplier',...
    'a3 multiplier','Si Repeat Length','Nano Repeat Length'};

%Extract just the important values from the matrix by eliminating the
%values for the three tests.
MatrixSub= Matrix(:,1:15);

%Prompt the user to save the data to file
ask=input('Do you want to write to MS EXCEL file(y/n)?   ','s');
if ask=='y'
filename=input('Name for file including extension(.xls):  ','s');
xlswrite(filename, label,1,'A1')
xlswrite(filename, MatrixSub,1,'A2')
end
```

B.2 SCLattice6.m

```
% SCLattice6 is a Matlab m-file written to create a repeatable unit
% cell mostly in the first quadrant with a hole drilled for a nanotube.
% Basic Process:
% First creates a silicon lattice centered at r=(0,0,0).
% Rotate the lattice so a chosen direction lies along the z-axis.
% Find two other lattice vectors that are nealy orthoganal to the
% chosen direction and to each other, includes a scaling
% factor for how close the perpedicular componant to the given
% direction is to a provided value. Use the three vectors to
% define a parallelepiped. Find an ion location that puts the center
% of the parallelpiped at the origin. Using that point, find
% the verticies of the parallelepiped. Remove any ions outside the
% parallelepiped. Remove any ions within a given radius of
% the z-axis. Make all the coordinates positive.

clc; clear all;
% Define values used in throughout the progam. Use the same scale for
% all values.
NUC=5;          % Number of unit cells to repeat in the X, Y and Z
                % directions Note that a smaller number speeds up
                % computation, but too small a number might not
                % include all the needed ions.
a0SC=5.4307;    % Define the lattice constant to be used in lattice
                % creation and for finding roughly orthoganal vectors
a1=[a0SC/2,a0SC/2,0]; % Give three primative lattice directions
a2=[a0SC/2,0,a0SC/2]; % to be used in finding roughly orthoganal
a3=[0,a0SC/2,a0SC/2]; % vectors.
xy=20;          % Give a optimal size for the perpedicular componant
                % of the roughly orthogonal vectors
mm=4;           % Give the maximum primitive vector multiplier
TL=11.52025439;     % Give the new z repeat distance
L=[0    0.707106781     0.707106781]; % Chosen vector to become the
                                      % z-axis
ntr=0;      % Give the nanotube radius
sep=0;      % Give the separation between the silicon and the nanotube

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Start Program
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Define a z-axis vector
ZZ=[0,0,1];

%Create a rotation matrix to put the lattice repeat diretion
%along the z-axis
angle1=acos(dot(L,ZZ)/(norm(L)*norm(ZZ)));
```

```
CP = cross(L,ZZ);
CPN = CP/norm(CP);
c = cos(angle1);
s = sin(angle1);
R = [CPN(1)^2+(1-CPN(1)^2)*c,CPN(1)*CPN(2)*(1-c)-CPN(3)*s,CPN(1)*...
    CPN(3)*(1-c)+CPN(2)*s;CPN(1)*CPN(2)*(1-c)+CPN(3)*s,CPN(2)^2+...
    (1-CPN(2)^2)*c,CPN(2)*CPN(3)*(1-c)-CPN(1)*s;CPN(1)*CPN(3)*(1-c)-...
    CPN(2)*s,CPN(2)*CPN(3)*(1-c)+CPN(1)*s,CPN(3)^2+(1-CPN(3)^2)*c;];

%Define X,Y,Z vectors
 X=[single(1) 0 0];
 Y=[0 single(1) 0];
 Z=[0 0 single(1)];

%Silicon Unit Cell Length
UL=a0SC;

%Unit Cell Atom locations
SiUnit(1,:)=[0 0 0];
SiUnit(2,:)=[UL/2 0 UL/2];
SiUnit(3,:)=[0 UL/2 UL/2];
SiUnit(4,:)=[UL/2 UL/2 0];
SiUnit(5,:)=[UL/4 UL/4 UL/4];
SiUnit(6,:)=[3*UL/4 UL/4 3*UL/4];
SiUnit(7,:)=[UL/4 3*UL/4 3*UL/4];
SiUnit(8,:)=[3*UL/4 3*UL/4 UL/4];

%Create the block of silicon
N=length(SiUnit);
count=1;
SiPreRotate=zeros((2*NUC)^3*8,3);
for z=-NUC-1:NUC;
 for y=-NUC-1:NUC;
    for x=-NUC-1:NUC;
     for n=1:N
        SiPreRotate(count,:)=SiUnit(n,:)+x*X*UL+y*Y*UL+z*Z*UL;
        count=count+1;
     end
    end
 end
end

% Apply the rotation matrix
l1=length(SiPreRotate);
SiAlmost=zeros(length(SiPreRotate),3);
for n=1:length(SiPreRotate);
    SiAlmost(n,:)=(R*SiPreRotate(n,:)')';
end

% Add the atomic number to front of each row
```

```
an=14;
AN=an+0*SiAlmost(:,1);
SCMatrix=horzcat(AN,SiAlmost);

% Rotate the lattice vectors and the direction vector
Lp=(R*L')';
a1p=(R*a1')';
a2p=(R*a2')';
a3p=(R*a3')';

% Find two other repeat directions using rotated lattice vectors
                % Create an array of possible vectors
                i=1;
                v=zeros(mm^3,3);
                for j=-mm:mm
                    for k=-mm:mm
                        for l=-mm:mm
                             if j==0 && k==0 && l==0
                                hell0=1;
                            else
                            v(i,1:3)=j*a1p+k*a2p+l*a3p;
                            i=i+1;
                            end
                        end
                    end
                end

                % Calculate the Cos of the angle between L and the
                % vectors in our array. If the Cos is negative, we set
                % the Cos and Size values to a large numbers so that
                % vector is ignored as there is another vector in the
                % matrix in exactly the opposite direction.
                mL=norm(Lp);
                l=size(v);
                DP=zeros(l(1,1),1);
                Cos=zeros(l(1,1),1);
                Size=zeros(l(1,1),1);
                for n=1:l
                    DP(n,1)=dot(v(n,1:3),Lp);
                    cos=abs(DP(n,1)/(mL*abs(norm(v(n,:)))));
                    if cos > 0
                    Cos(n,1)=cos;
                    Size(n,:)=abs(norm(v(n,:)-DP(n,1)/mL^2*Lp)-xy);
                    else
                    Cos(n,1)=100;
                    Size(n,:)=100;
                    end
                end

                % Compile the values calculated into a matrix with the
```

```
% vectors.
sort1=horzcat(v,Cos,Size);

% Calculate a value factor for each vector
sort1(:,6)=10*sort1(:,4)+sort1(:,5);

% Sort the vectors by the value factor and eliminate
% duplicates
matrix1=sortrows(unique(sort1,'rows'),6);

% Grab the first 50 and check them for orthoganality
% to each other.
vs=matrix1(1:100,:);
i=1;
testv=zeros(100,3);
vt1=zeros(100^2,3);
vt2=zeros(100^2,3);
Cos12=zeros(100^2,1);
Cos13=zeros(100^2,1);
Cos23=zeros(100^2,1);
DP2=zeros(100^2-1,1);
size1=zeros(100^2,1);
size2=zeros(100^2,1);
for p=1:100
    testv(p,:)=vs(p,1:3);
    mp=norm(testv(p,:));
    for n=1:100
        vt1(i,:)=testv(p,:);
        vt2(i,:)=vs(n,1:3);
        if p==n
            Cos12(i,1)=100;
            Cos13(i,1)=100;
            Cos23(i,1)=100;
            size1(i,1)=100;
            size2(i,1)=100;
        else
            DP2(i,1)=vt1(i,1)*vt2(i,1)+vt1(i,2)...
                *vt2(i,2);
            Cos12(i,1)=vs(p,4);
            Cos13(i,1)=vs(n,5);
            Cos23(i,1)=abs((vt1(i,1)*vt2(i,1)+...
                vt1(i,2)*vt2(i,2))/(sqrt(vt1(i,1)^2+...
                vt1(i,2)^2)*sqrt(vt2(i,1)^2+...
                vt2(i,2)^2)));
            size1(i,:)=matrix1(p,5);
            size2(i,:)=matrix1(n,5);
        end
        i=i+1;
    end
end
```

```
                % Compile a Matrix using the values calulated above
                sort2=horzcat(vt1,vt2,Cos12,Cos13,Cos23,size1,size2);

                % Calculate a total value factor
                sort2(:,12)=10*sort2(:,7)+10*sort2(:,8)+10*...
                    sort2(:,9)+sort2(:,10)+sort2(:,11);

                % Sort the vectors by the value factor
                matrix2=sortrows(unique(sort2,'rows'),12);

                % Extract the found vectors
                v1=matrix2(1,1:3);
                v2=matrix2(1,4:6);

% Define a vector that will be used as location
% to start the parallelepiped
v3=[0 0 TL];
P=-0.47*(v1+v2+v3);

% Remove any ions that are outside the parallelepiped
l=length(SCMatrix);
i=1;
v4=v1+v2+v3;
P4=P+v4;
vp1=cross(v1,v2);
vp2=cross(v2,v3);
vp3=cross(v3,v1);
c1=sign(dot(vp1,v4));
c2=sign(dot(vp2,v4));
c3=sign(dot(vp3,v4));
c4=sign(dot(vp1,-v4));
c5=sign(dot(vp2,-v4));
c6=sign(dot(vp3,-v4));
while i<=l
    vc1=SCMatrix(i,2:4)-P;
    vc2=SCMatrix(i,2:4)-P4;
    cc1=sign(dot(vp1,vc1));
    cc2=sign(dot(vp2,vc1));
    cc3=sign(dot(vp3,vc1));
    cc4=sign(dot(vp1,vc2));
    cc5=sign(dot(vp2,vc2));
    cc6=sign(dot(vp3,vc2));
    if abs(c1-cc1)+abs(c2-cc2)+abs(c3-cc3)+abs(c4-cc4)+...
            abs(c5-cc5)+abs(c6-cc6) ~= 0
        SCMatrix(i,:)=[];
        i=i-1;
        l=l-1;
    end
    i=i+1;
```

```
end

% Remove atoms within r of the z axis
r=ntr+sep;
l=length(SCMatrix);
i=1;
while i<=l
    if sqrt((SCMatrix(i,2).^2+SCMatrix(i,3).^2))<=r
        SCMatrix(i,:)=[];
        i=i-1;
        l=l-1;
    end
    i=i+1;
end

% Check the order of the basis vectors because VASP needs the vectors
% in an order that gives a positive triple product
if dot(v1,cross(v2,v3))<0
    v1tp=v2;
    v2tp=v1;
    v1=v1tp;
    v2=v2tp;
end

%Shift Lattice so it is in the "First" quadrant
mx=min(SCMatrix(:,2));
my=min(SCMatrix(:,3));
mz=min(SCMatrix(:,4));
MV=horzcat(-mx,-my,-mz);
Shift=repmat(MV,length(SCMatrix),1);
SCMatrix(:,2:4)=SCMatrix(:,2:4)+Shift;

% Output atom locations to xyz or POSCAR file. File is output in
% folder where SCLattic6 is located
ask=input('Do you want to write to xyz file(y/n)?   ','s');
if ask=='y'
filename=input('Name for file including extension:  ','s');
note=input('Note for second line of xyz file:  ', 's');
dlmwrite(filename, length(SCMatrix), 'delimiter', ' ','newline','pc')
dlmwrite(filename, note,'-append', 'delimiter', '','newline', 'pc',...
    'roffset', 0)
dlmwrite(filename, SCMatrix, '-append', 'delimiter', '\t',...
    'precision', 8, 'newline', 'pc','roffset', 0)
end

ask=input('Do you want to write to VASP POSCAR file(y/n)?   ','s');
if ask=='y'
temp=SCMatrix(1:length(SCMatrix),2:4);
filename=input('Name for file including extension:  ','s');
note=input('Note for first line of POSCAR file:  ', 's');
```

```
dlmwrite(filename, note,'delimiter', '','newline', 'pc')
dlmwrite(filename, 1, '-append', 'delimiter', '','newline', 'pc',...
    'roffset', 0)
dlmwrite(filename, v1, '-append', 'delimiter', '\t', 'precision',...
    '%1.6f', 'newline', 'pc','roffset', 0)
dlmwrite(filename, v2, '-append', 'delimiter', '\t', 'precision',...
    '%1.6f', 'newline', 'pc','roffset', 0)
dlmwrite(filename, v3, '-append', 'delimiter', '\t', 'precision',...
    '%1.6f', 'newline', 'pc','roffset', 0)
dlmwrite(filename, length(SCMatrix),'-append', 'delimiter', '',...
    'newline', 'pc','roffset', 0)
dlmwrite(filename, 'Cartesian', '-append', 'delimiter', '',...
    'newline', 'pc','roffset', 0)
dlmwrite(filename, temp,'-append', 'delimiter', '\t', 'precision',...
    '%1.6f', 'newline', 'pc')
end
```

B.3 VASPDOSPlot.nb

```
(*VASPDOSPlot is a Mathematica Module that uses system dialogs to
generate quality DOS plots from VASP DOSCAR files.  It allows the user
to select the file using a system dialog window.  It includes labels and
calculates the resolution of the DOS plot*) Module[{}, (*Open a dialog

window that lets the user select the DOSCAR file to import*)
filename=SystemDialogInput["FileOpen", NotebookDirectory[],
WindowTitle- >"Select DOSCAR File"]; (*Create an error dialog if no file

is selected*)
If[filename==$Canceled, Print[Style["Error:No File Slected", 24, Red,
Bold]]; Abort[]];

(*Inport the DOSCAR file separaring the data into useful sections*)
fid=OpenRead[filename];
(*Inport header data*)
header1=Read[fid, {Number, Number, Number, Number}];
header2=Read[fid, {Number, Number, Number, Number, Number}];
header3=Read[fid, {Number}];
header4and5=Read[fid,{String, String}];
header6=Read[fid, {Number, Number, Number, Number, Number}]; (*Inport
DOS data*)
DOSdataRAW=ReadList[fid, {Number, Number, Number}, header6[[3]]];

(*Manipulate inported data*)
DOSdataTemp=Transpose[DOSdataRAW];
DOSdata=Transpose[{DOSdataTemp[[1]], DOSdataTemp[[2]]}];

(*Calculate the resolution of the DOS*)
DOSresolution=ToString[Evaluate[Abs[DOSdataTemp[[1]][[1]] -
```

```
DOSdataTemp[[1]][[2]]]]];

(*Create a lable that includes the DOS resolution*)
ylabel=StringJoin["DOS"," (number of states per ", DOSresolution,"eV)"];

(*Extract the Fermi-Level*)
FermiLevel=header6[[4]];

lines=Graphics[{Dashed,Thickness->0.001,
Line[{{FermiLevel,0},{FermiLevel, 1.05*Max[DOSdataTemp[[2]]]}}]}];
(*Plot the DOS*)
plot=ListPlot[DOSdata,Joined->True,Frame->True,PlotRange->
{{Min[DOSdataTemp[[1]]], Max[DOSdataTemp[[1]]]},
{0,1.05*Max[DOSdataTemp[[2]]]}}, AxesOrigin->{Min[DOSdataTemp[[1]]], 0},
FrameLabel->{"Energy(eV)", ylabel}, AspectRatio->2/3,
ImageSize->1000, LabelStyle->{32,FontFamily->"Trebuchet MS"},
FrameTicksStyle->Directive[Black, Large], ImagePadding->All,
PlotStyle->{Thick, Blue}];
Show[plot, lines]
]
```

B.4 VASPBandPlot.nb

```
(*VASPBandPlot is a Mathmatica Module that uses interactive dialogs to
generate quality band structure plots from VASP EIGENVAL files.  It
allows the user to select the file using a system dialog window.  It
prompts the user to imput the fermi energy which can be found in the
DOSCAR or OUTCAR file but is not included in the EIGENVAL file.  It uses
an interactive window to step the user through labeling the critical
points.  It then generates two plots, one of the complete band structure
and one of the band structure near the fermi energy.  These plots have
slider bars to adjust the aspect ratio and the the size of the plots.
The program is desigened to be self-contained which allows users who are
not familiar with Mathematica to use it and makes it possible to load it
as a input executable.*)

Module[{},
(*Open a dialog window that lets the user select the EIGENVAL file to
import*)
filename=SystemDialogInput["FileOpen", NotebookDirectory[],
WindowTitle->"Select EIGENVAL File"];

(*Open a dialog window that allows the user to input the fermi energy.
This will be used to create a zoomed-in graph of the band structure near
the fermi energy.*)
FermiEnergy=Input["Please enter the fermi energy", WindowTitle->"Fermi
energy", FieldSize->Tiny, WindowSize->All];
```

```
(*Create an error dialog if no file is selected*)
If[filename==$Canceled, Print[Style["Error:No File Slected", 24, Red,
Bold]]; Abort[]];

(*Inport the EIGENVAL file separaring the data into useful sections*)
fid=OpenRead[filename];
(*Inport header data*)
Line1Raw=Read[fid, {Number, Number, Number, Number}];
Line2Raw=Read[fid, {Number, Number, Number, Number, Number}];
Line3Raw=Read[fid, {Number}];
SystemNameRaw=Read[fid, {String, String}];
FourTimesIons=Read[fid, {Number}][[1]];
NumKpoints=Read[fid, {Number}][[1]];
NumBands=Read[fid, {Number}][[1]];
(*Initialize variables used to store the kpoints and the bands*)
KpointsRaw=Table[0, {NumKpoints}];
BandsRaw=Table[0, {NumKpoints}];
(*Inport kpoints and bands*)
For[ i=1, i<=NumKpoints, i++, KpointsRaw[[i]]=Read[fid, {Number, Number,
Number, Number}];BandsRaw[[i]]=ReadList[fid, {Number, Number},
NumBands]];

(*Extract the bands from the imported data*)
Bands=Transpose[BandsRaw, {3, 2, 1}][[2]];

(*Create a list of the band energies to be used in selecting plot
ranges*)
Energies=Union[Flatten[Bands]];

(*Open a dialog window that will allow the user to select the number of
kpoints in the brillouin zone path and assign symbols to the critical
points*)
CreateDialog[{Label[reset];

(*Initialize variables used in the dialog and turn off a unneeded
error*)
numCP=0;
Clear["CP1", "CP2", "CP3", "CP4", "CP5", "CP6", "CP7", "CP8", "CP9",
"CP10", "CP11", "CP12"];Off[General::iterb];

(*This manipulate allows the user to modify both the number of critical
points and the critical point labels*)
Manipulate[

(*Create a switch that requires the user to select the number of
critical points before assigning symbols*)
If[numCP==0, " ",

(*Create a set of tabs and radio-buttons that can be used to assign
symbols to the critical points*)TabView[Table[With[{nnn=nnn},
```

```
RadioButtonBar[Dynamic[Evaluate[Symbol["CP"<>ToString[nnn]]]], {"Γ",
"A", "B", "C", "D", "F", "H", "K", "L", "M", "N", "P", "R", "S", "T",
"U", "W", "X", "Y", "Z"}, Appearance− >Large, BaselinePosition− >Bottom,
LabelStyle− >{32, FontFamily− >"Trebuchet MS"}]], {nnn, 1, numCP, 1}]]],

(*Create a plot in the dialog window that can be used to check that the
assigned symbols are correct.  The first plot contains the bands.  The
second plot contains the labels.  This is required because ListPlot will
not label the Tick at the origin.*)
Dynamic[pl1=ListPlot[Bands, Joined− >True, Frame− >True,
PlotRange− >{{1, NumKpoints}, {FermiEnergy-2, FermiEnergy+2}},
AxesOrigin− >{0, FermiEnergy}, AxesStyle− >Thick, FrameStyle− >Thick,
AspectRatio− >16/9, ImageSize− >400, FrameTicks− >None,
ImagePadding− >All, PlotStyle− >{{Thick, Blue}}];
pl2=Plot[, {x, 0, NumKpoints}, Frame− >True, FrameStyle− >Thick,
FrameTicksStyle− >Directive[Black, 16], FrameTicks− >If[numCP==2, {{{1,
CP1}, {NumKpoints/2, ""}, {NumKpoints, CP2}}, Automatic},
{Table[With[{n=n}, {1+(NumKpoints-1)/(numCP-1)*(n-1),
Symbol["CP"<>ToString[n]]}], {n, 1, numCP, 1}], Automatic}],
FrameLabel− >{None, Style["Energy(eV)", 15]}, LabelStyle− >{16,
FontFamily− >"Trebuchet MS"}, AxesStyle− >Thick,
AxesLabel− >{Style["Fermi Level", 8, FontFamily− >"Trebuchet MS", Bold],
None}, PlotRange− >{{1, NumKpoints}, {FermiEnergy-2, FermiEnergy+2}},
AxesOrigin− >{0, FermiEnergy}, AspectRatio− >3/4, ImageSize− >400,
ImagePadding− >All];
Show[{pl2, pl1}, Background− >White]], {{numCP, 12, "Number of Critical
Points"}, 2, 12, 1, ControlType− >PopupMenu, BaseStyle− >{Bold, 16,
FontFamily− >"Trebuchet MS"}}, LabelStyle− >{Bold, 16,
FontFamily− >"Trebuchet MS"}, AppearanceElements− >None,
Paneled− >False, LocalizeVariables− >False], (*Turn the error back on*)

On[General::iterb];

(*Create two buttons.  The first is just an OK button.  The second
resets all the parameters and allows the user to restart the dialog.
This is needed because if the number of critical points is changed the
symbols assigned to the critical points cannot be modified*)
Row[{DefaultButton[ImageSize− >{75, Medium}], Button["Reset",
numCP=0;Clear["CP1", "CP2", "CP3", "CP4", "CP5", "CP6", "CP7", "CP8",
"CP9", "CP10", "CP11", "CP12"];, ImageSize− >{75, Medium}]}, " "]},
WindowTitle− >"Select Critical Point Labels", WindowSize− >{72 7, 72 8},
Modal− >True];
Print[Style["Adjust the parameters until the plots look good", 32,
FontFamily− >"Trebuchet MS", Bold]];

(*Create a Manipulate that can be used to adjust the size and
aspect-ratio of the generated plots*)
Manipulate[

(*Create a complete plot of the entire band structure.  The first plot
```

```
contains the bands.  The second plot contains the lables.  This is
required because ListPlot will not label the Tick at the origin.*)
pl3=ListPlot[Bands, Joined− >True, Frame− >True, PlotRange− >{{1,
NumKpoints}, {1.05*Min[Energies], 1.05*Max[Energies]}}, AxesOrigin− >{1,
1.05*Min[Energies]}, FrameLabel− >None, FrameStyle− >Thick,
FrameTicks− >None, AspectRatio− >aspectratio, ImageSize− >imagesize,
ImagePadding− >All, PlotStyle− >{Thick}];
pl4=Plot[, {x, 1, NumKpoints}, Frame− >True, PlotRange− >{{1,
NumKpoints}, {1.05*Min[Energies], 1.05*Max[Energies]}}, AxesOrigin− >{1,
1.05*Min[Energies]}, FrameLabel− >{None, "Energy(eV)"},
FrameStyle− >Thick, FrameTicksStyle− >{32, 48},
FrameTicks− >If[numCP==2, {{{1, CP1}, {NumKpoints/2, ""}, {NumKpoints,
CP2}}, Automatic}, {Table[With[{n=n}, {1+(NumKpoints-1)/(numCP-1)*(n-1),
Symbol["CP"<>ToString[n]]}], {n, 1, numCP, 1}], Automatic}],
FrameLabel− >{None, Style["Energy(eV)", 32]}, LabelStyle− >{32,
FontFamily− >"Trebuchet MS"}, AxesStyle− >Thick,
AspectRatio− >aspectratio, ImageSize− >imagesize, ImagePadding− >All];

(*Create a zoomed-in plot of the band structure near the fermi energy.
The first plot contains the bands.  The second plot contains the lables.
This is required because ListPlot will not lable the Tick at the
origin.*)
pl5=ListPlot[Bands, Joined− >True, Frame− >True, PlotRange− >{{1,
NumKpoints}, {FermiEnergy-2, FermiEnergy+2}}, AxesOrigin− >{1,
FermiEnergy}, AxesStyle− >Thick, FrameLabel− >{None, Style["Energy(eV)",
32]}, FrameStyle− >Thick, LabelStyle− >{32, FontFamily− >"Trebuchet
MS"}, AspectRatio− >AspectRatio, ImageSize− >imagesize,
FrameTicks− >Automatic, FrameTicksStyle− >Directive[Black, 32],
ImagePadding− >All, PlotStyle− >{{Thick, Blue}}];
pl6=Plot[, {x, 1, NumKpoints}, Frame− >True, PlotRange− >{{1,
NumKpoints}, {FermiEnergy-2, FermiEnergy+2}}, AxesOrigin− >{1,
FermiEnergy}, AxesStyle− >Thick, FrameLabel− >{None, "Energy(eV)"},
FrameStyle− >Thick, FrameTicksStyle− >{32, 48},
FrameTicks− >If[numCP==2, {{{1, CP1}, {NumKpoints/2, ""}, {NumKpoints,
CP2}}, Automatic}, {Table[With[{n=n}, {1+(NumKpoints-1)/(numCP-1)*(n-1),
Symbol["CP"<>ToString[n]]}], {n, 1, numCP, 1}], Automatic}],
FrameLabel− >{None, Style["Energy(eV)", 32]}, LabelStyle− >{30,
FontFamily− >"Trebuchet MS"}, AxesLabel− >{Style["Fermi Level", 16,
FontFamily− >"Trebuchet MS", Bold], None}, AspectRatio− >aspectratio,
ImageSize− >imagesize, ImagePadding− >All];

(*Create dashed vertical lines at the critical points*)
lines=Graphics[{Dashed, Thickness− >0.001, Line[Table[With[{n=n},
{{1+(NumKpoints-1)/(numCP-1)*(n-1), Min[Energies]},
{1+(NumKpoints-1)/(numCP-1)*(n-1), Max[Energies]}}], {n, 1, numCP,
1}]]}];

(*Show both the plots*)
Column[{Show[{pl4, pl3, lines}], Show[{pl6, pl5, lines}]}],
{{aspectratio, 3/4, "AspectRatio"}, 1/4, 2, 1/16}, {{imagesize, 500,
```

```
"ImageSize"}, 100, 2000, 100}, LabelStyle− >{Bold, 16,
FontFamily− >"Trebuchet MS"}]
]
```

Appendix C

Characterization of Silicon

The first DOS plots I produced for my systems looked like a bunch of noise. I was unsure if that was really what the DOS looked like or if there was something wrong. I decided to explore my settings for VASP using silicon. I chose silicon because its DOS is well known and it has only two atoms in its primitive unit cell so my computations could run quickly. Also, for all of these runs I chose to not allow the atomic locations to relax to reduce computation. Out of curiosity, I performed a run relaxation run for the silicon primitive unit cell and found that the original atom locations were relaxed, suggesting that the relaxation step is not necessary for crystalline silicon.

The settings for VASP can be changed in two ways. The first is changing the number and location of the k-points. Since I was primarily interested in how the number of k-points affects the quality of the DOS plots, that was the first place I started. I chose the automatic k-point generation scheme in VASP and just varied the number of k-points. The second way to change the settings is by setting keys in one of the VASP input files. These keys affect, among other things, the energy cutoffs and the resolution of the DOS generated.

The accepted density of states for silicon is shown in Fig. C.1. This is what I am

comparing my results to and hopefully can match with the correct settings.

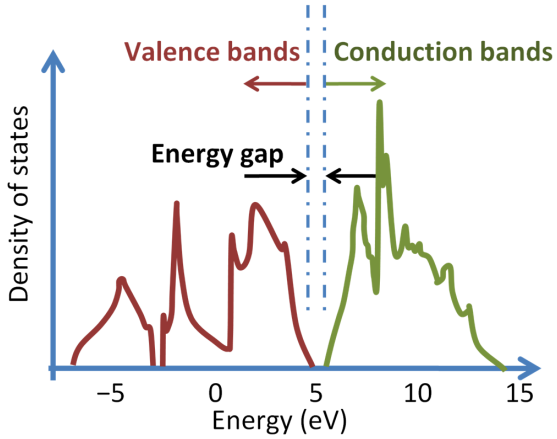


Figure C.1 Accepted density of states for crystalline silicon.

C.1 Silicon Primitive Unit Cell

I started with the primitive unit cell because it only contains two atoms and the calculations can be done really quickly. In addition, I can use really fine k-point meshes that would be too computationally intensive for larger systems. I started off with a mesh of twenty k-points in each reciprocal lattice direction, well above what I had been using for my silicon-nanotube systems, Fig. C.2 The results looked fairly close to Fig. C.1, but there is a little noise in the peak at around 4 eV.

I wanted to see if a finer k-mesh would look better so I used a 38x38x38 k-mesh which was the finest k-mesh the compiled version of VASP I was using would allow Fig. C.3 Using a higher k-mesh produced a nearly perfect silicon density of states, assuring me that it was possible to get good results using my method with really fine k-meshes.

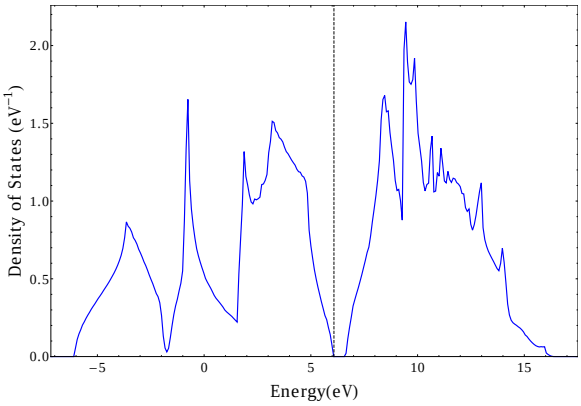


Figure C.2 Calculated DOS for silicon primitive unit cell with a 20x20x20 k-mesh. The dashed line represents the Fermi-level.

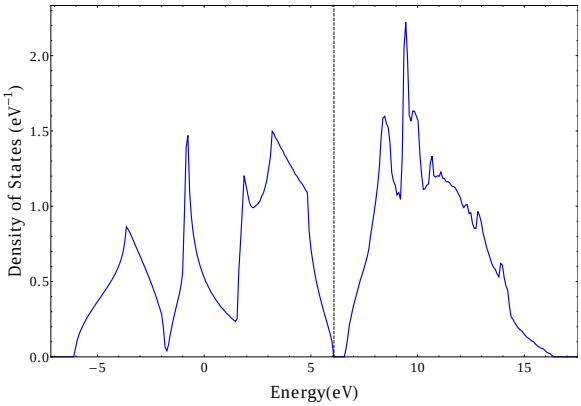


Figure C.3 Calculated DOS for silicon primitive unit cell with a 38x38x38 k-mesh.

C.2 Silicon Conventional Unit Cell

Having obtained a high quality band structure for the primitive unit cell I decided to explore with the conventional unit cell which contains eight atoms. My initial though was that because the unit cell was bigger, the Brillouin zone would be smaller which would mean that I could get away with fewer k-points. Using a 38x38x38 k-mesh for the conventional unit cell shows that this is not true Fig. C.4. This DOS plot has even more noise that the primitive unit cell with a 20x20x20 k-mesh Fig. C.2.

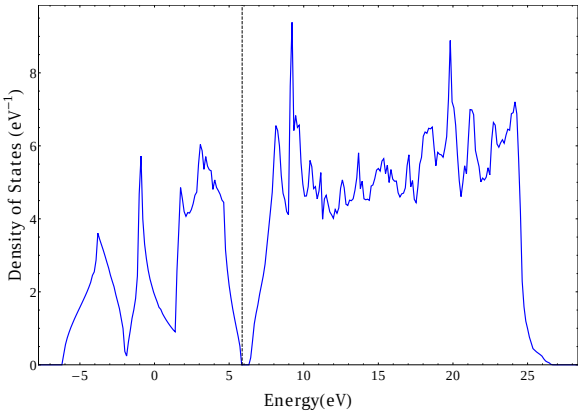


Figure C.4 Calculated DOS for silicon conventional unit cell with a 38x38x38 k-mesh.

Another thing to notice about this plot is that the peak above the band-gap is wider. I am not entirely sure why that it appears the way it does, but it might be related to VASP using more bands with a higher number of atoms.

There is a tag in VASP(NEDOS) that lets you specify the resolution of the DOS. I used a 30x30x30 k-mesh and increased the resolution from 300 points to 5,000 points to see if anything changed Fig. C.5. Very little changed by increasing the resolution of

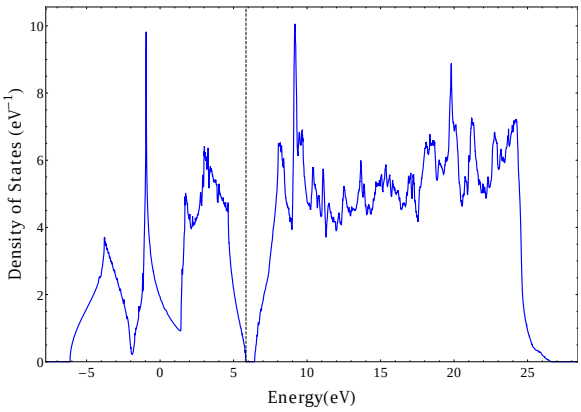


Figure C.5 Calculated DOS for silicon conventional unit cell with a 30x30x30 k-mesh and 5,000 point resolution.

the DOS except that the noise became more well defined. I concluded that increasing

the resolution does not produce better looking DOS plots.

C.3 Silicon Conventional Unit Cell Repeated

My unit cells are much larger than eight atoms so I chose to create larger silicon unit cells by replicating the conventional unit cell to two, four and eight times to see if I could find any trend in moving to larger systems.

C.3.1 Two Conventional Unit Cells

VASP has a feature that will automatically generate the k-mesh for you. I used this method when doing my computations and wanted to see if using the automatically generated mesh produced accurate results. VASP produced a 9x18x18 k-mesh with the 9 corresponding to the direction of the replication Fig. C.6 This DOS continued

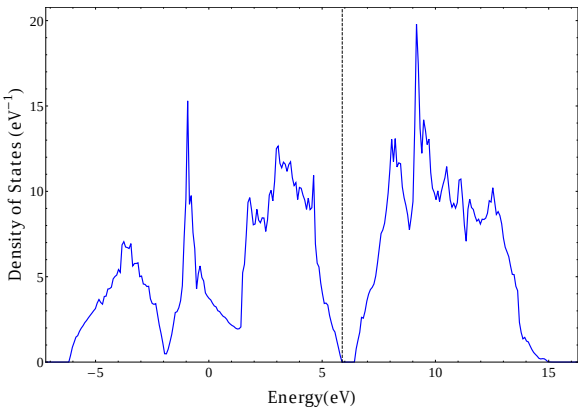


Figure C.6 Calculated DOS for silicon conventional unit cell replicated to two times with a 9x18x18 k-mesh.

the trend of more noise with larger unit cells.

C.3.2 Four Conventional Unit Cells

I replicated the conventional unit cell in the x direction and then the y direction to obtain four unit cells. I used the automatic generation scheme which produced a 9x9x18 k-mesh Fig. C.7. Once again we see more noise with a larger unit cell.

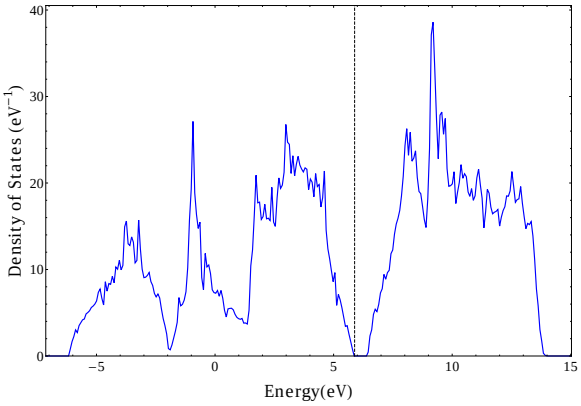


Figure C.7 Calculated DOS for silicon conventional unit cell replicated to four times with a 9x9x18 k-mesh.

C.3.3 Eight Conventional Unit Cells

I replicated the conventional unit cell in the x then y then z directions producing a supercell containing eight conventional unit cells. This was the closest in size to my nanotube-silicon unit cells. I initially started with the automatic generation(9x9x9 k-mesh) which produced a really noisy DOS Fig. C.8.

I tried a finer k-meshes(20x20x20), but could not eliminate the noise Fig. C.9, convincing me that larger unit cells do not need fewer k-points to produce good looking DOS plots using VASP.

One of the settings you can specify in VASP is how to handle partially occupied density of states. Up until now I had been using tetrahedral smearing(ISMEAR=-5) as was recommended for producing good DOS plots in the VASP manual. In

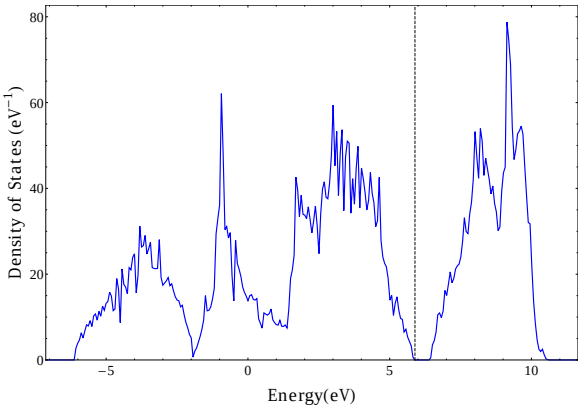


Figure C.8 Calculated DOS for silicon conventional unit cell replicated to eight times with a 9x9x9 k-mesh.

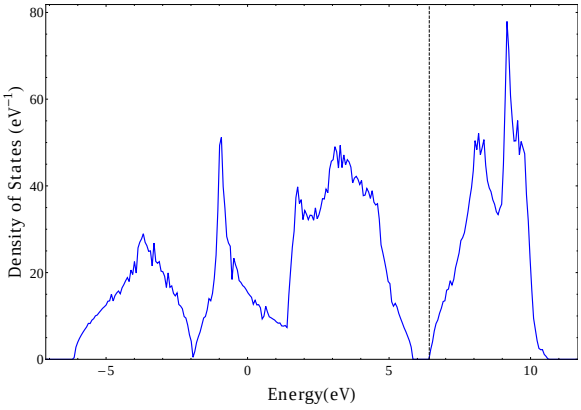


Figure C.9 Calculated DOS for silicon conventional unit cell replicated to eight times with a 20x20x20 k-mesh.

another part of the manual Gaussian smearing(ISMEAR=0) with a small sigma was recommended for large unit cells. I tried Gaussian smearing with sigma equal to .1eV Fig. C.10.

Using Gaussian smearing produced a DOS that looks very much like the accepted DOS. In addition, this DOS looks like it has undergone some smoothing which eliminated any sharp features of the DOS. In particular, the gap region is very rounded, limiting the information we can obtain from the DOS plot. We are unsure how the handling of partially occupied states affects the entire DOS in VASP. The manual

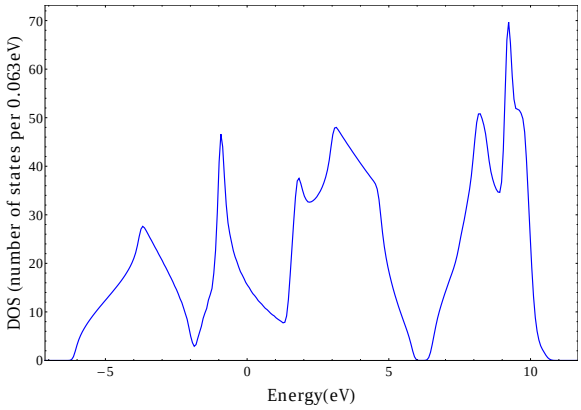


Figure C.10 Calculated DOS for silicon conventional unit cell replicated to eight times with a 20x20x20 k-mesh and Gaussian smearing with sigma=0.1eV.

does not give any clear explanations, which is why had to explore the settings using silicon.

As I mentioned before, when using Gaussian smearing, you have to specify a sigma value for the smearing. I had hoped that by increasing sigma, you could obtain sharper features. I tried Gaussian smearing with sigma equal to 0.05eV Fig. C.11, and found that a smaller sigma produced noise, but a different kind of noise than when using tetrahedral smearing. This noise was more wavelike and tended to be more pronounced at the where two peaks meet.

After playing around a bit more, I found that I could produce a DOS plot similar to Fig. C.10 using an 8x8x8 k-mesh, Gaussian smearing, and sigma equal to 0.2eV Fig. C.12.

C.4 Silicon With a Hole

Lastly, I took the eight conventional unit cells and removed eighteen atoms to create a hole in the center of a supercell that runs along the z-axis. These supercells are

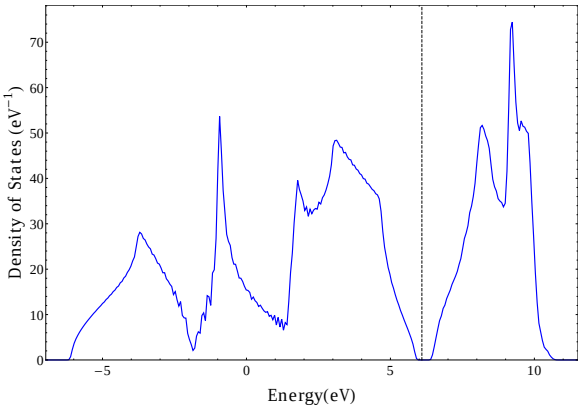


Figure C.11 Calculated DOS for silicon conventional unit cell replicated to eight times with a 20x20x20 k-mesh and Gaussian smearing with sigma=0.05eV.

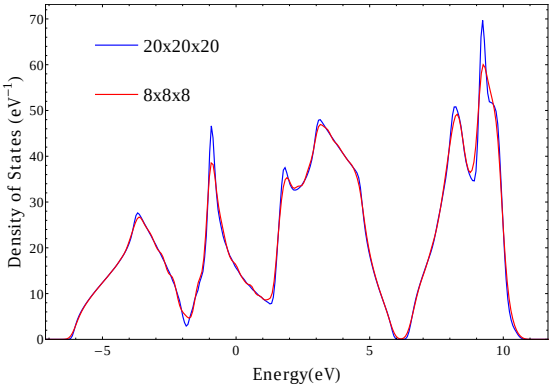


Figure C.12 Comparison of calculated DOS for 20x20x20 and 8x8x8 k-meshes with sigma=0.1 and sigma=0.2 respectively.

smaller versions of the silicon portion of my nanotube-silicon systems, which would give me a good idea of the settings to use. I first used the same settings as Fig. C.10, 20x20x20 k-mesh and Gaussian smearing with sigma equal to 0.1eV Fig. C.13

This density of states looked pretty good, but I wanted to make sure that sigma equal to 0.1eV was the best choice. I tried the same job but with sigma equal to 0.5eV Fig. C.14 and 0.2eV C.15.

The smaller sigma produced the noise I had seen earlier and the larger sigma

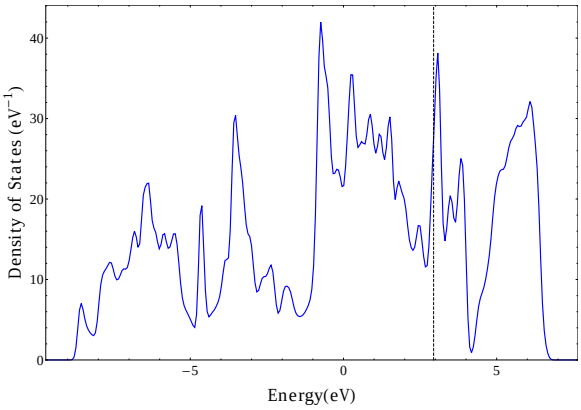


Figure C.13 Calculated DOS for a silicon supercell with a hole, 20x20x20 k-mesh and Gaussian smearing with sigma=0.1eV .

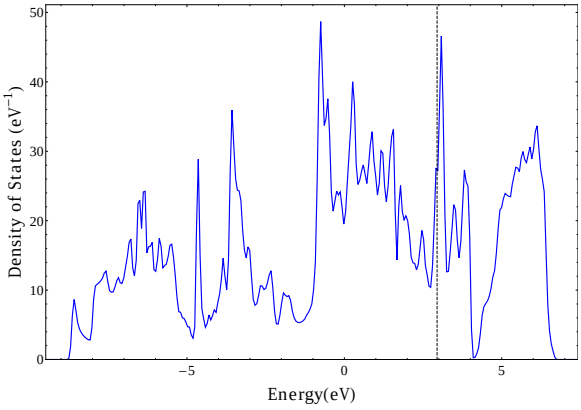


Figure C.14 Calculated DOS for a silicon supercell with a hole, 20x20x20 k-mesh and Gaussian smearing with sigma=0.05eV .

seemed to smooth the DOS too much. I wanted to make sure that what I thought was noise was noise and not features of the actual DOS. I thought that by increasing the k-mesh I could see if anything changed. If nothing changed, then my DOS was good enough. I used a 30x30x30 k-mesh Fig. C.16 which produced a nearly identical DOS to Fig. C.13. With this result, I felt confident that Gaussian smearing(ISMEAR=0) and sigma equal to 0.1eV(SIGMA=0.1) produced the best DOS plots. Although, these settings smooth some of the DOS features, the lack of noise makes them the best choice. In addition, choosing the finest k-mesh feasible assures the most accurate

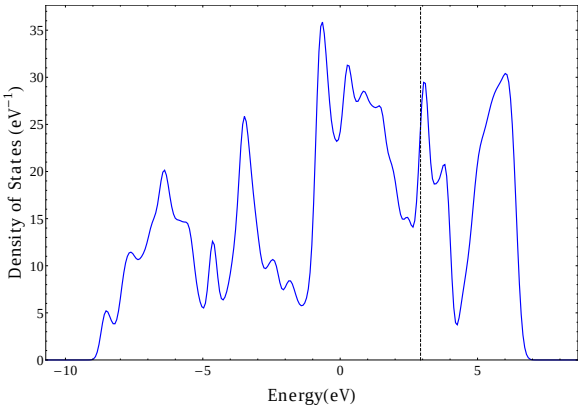


Figure C.15 Calculated DOS for a silicon supercell with a hole, 20x20x20 k-mesh and Gaussian smearing with sigma=0.2eV .

DOS plots.

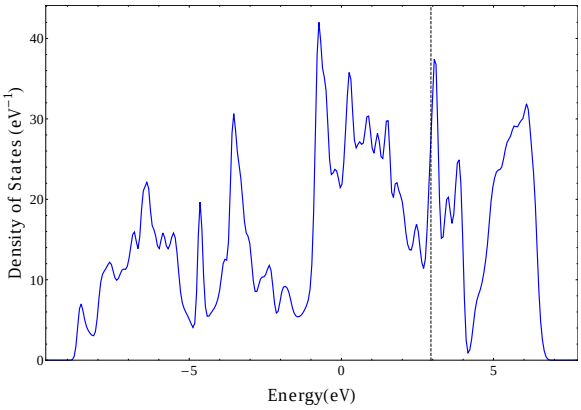


Figure C.16 Calculated DOS for a silicon supercell with a hole, 30x30x30 k-mesh and Gaussian smearing with sigma=0.1eV .