Selecting Semiconductor Materials For The Quantum Dot Intermediate Band Solar Cell

A Thesis
Submitted to the Faculty
of
Drexel University
by
Steven Evans Jenks
in partial fulfillment of the
requirements for the degree
of
Doctor of Philosophy
March 2012

Dedications

This thesis is dedicated to my wife, Jessica. Her patience, love, and support made this work possible.

Acknowledgments

I want to first thank my advisor, Dr. Robert Gilmore, for his willingness to spend the countless hours that made our research possible. His insight and wisdom uncovered the subtle problems that refined this work, his guidance and vision always kept me on the right path, and those Friday afternoon 'discussions' honed my ability to become a more effective physicist.

The entire physics department has helped me in countless ways throughout my entire career at Drexel. In particular, I would like to thank Dr. Michael Vogeley for taking a chance on me in 2004 by admitting an interested engineer and helping me navigate the 'part-time' student waters. I want to thank my colleagues Erica Caden, Danny Pan, Donna Yosmanovich, and Travis Hoppe for their encouragement and helpful correspondence.

I want to thank my parents, John and Linda Jenks, for giving me every opportunity to succeed, always encouraging excellence, and never allowing me to settle for mediocrity. They instilled the work ethic, integrity, and values that make up the person I am today. I want to thank my two brothers, John and Andy Jenks, for their support and teaching me all those 'things' that big brothers teach younger brothers. Lastly, I thank my wonderful wife Jessica for her unconditional love and providing laughter at moments when I needed it most.

Table of Contents

List of Tables

List of Figures

Abstract

Selecting Semiconductor Materials For The Quantum Dot Intermediate Band Solar Cell

Steven Evans Jenks

Robert Gilmore, Ph.D.

The main limitations of the conventional solar conversion device is that low energy photons cannot excite charge carriers to the conduction band, therefore do not contribute to the devices's current, and high energy photons are not efficiently used due to a poor match to the energy gap. However, if intermediate levels are introduced into the energy gap of a conventional solar cell, then low energy photons can be used to promote charge carriers in a stepwise manner to the conduction band thereby enhancing the current while maintaining a large open-circuit voltage. This concept is called the intermediate band solar cell and increases the efficiency beyond the thermodynamic limits of the conventional solar cell. A device based on the confined electron levels of quantum dots called the quantum dot intermediate band solar cell is proposed as a physical realization of the intermediate band solar cell. In this work, we propose material systems that are considered candidates for the quantum dot intermediate solar cell.

Need more text...

Chapter 1: Introduction and Background

According to the U.S. Energy Information Administration (EIA), the world's total energy consumption in 2007 was 495.2 quadrillion British Thermal Units (BTU) with 86% derived from fossil fuels[1] [2]. If all nonrenewable sources[2] are considered, there is an additional 6% from nuclear, bringing the world's total energy production from this category to 94%. The current asymmetries in the distribution of nonrenewable sources of energy is unsustainable, meaning we can assume with complete assurance that with the status quo of energy production, exploitation of nonrenewable resources will consist in the progressive exhaustion of an initially fixed supply in which there will be no significant additions. When will nonrenewables be exhausted? This is an outstanding question surrounded by numerous areas of debate. In 1956, M. King Hubbert explored this concept by realizing that there will be a point in time when the maximum rate of fossil fuel extraction will be reached and afterwards the rate of extraction enters a terminal decline until the finite resource is completely exhausted. The point in time when this maximum rate is reached just before the terminal decline is called "peak". He was the first to give rise to the term peak and develop extrapolating models predicting peak, using his models to accurately determine peak for U.S. oil production would occur between years of 1965 and 1970 [3]. Further exasperating the exploitation of nonrenewable resources is the fact that energy demand is increasing. By 2035, the world's energy consumption is projected to be 738.7 quadrillion BTU [2] or an increase of 49% from 2007. Our insatiable thirst for energy coupled with our current dependence on nonrenewables will cause devastating consequences if the pendulum of energy production doesn't shift dramatically towards renewables. Some of the more obvious consequences include conflicts revolving around securing diminishing resources, global warming and economic turmoil, not to mention the regression of human quality of life as we know it.

The amount of solar irradiance that hits the earth each year is approximately 763,000 quadrillion Btu[3] or over 1000 times more energy than what human energy consumption is projected to be 2035. If a tiny fraction of energy that earth receives could be converted to useful energy, all our energy supply problems would be solved. So what is preventing us from tapping into this seemingly endless amount of energy that falls on the earth everyday to solve the scarcity, environmental, and economic problems associated with nonrenewables? The two main market penetration barriers of solar conversion devices include the current state of technology and economics, both intertwined.

Today, the majority of solar conversion devices make electricity available as the end use form of energy by using either concentrating solar thermal technologies or photovoltaic cells. Concentrating thermal solar technologies utilize the sun as the heat source to boil water into steam which in turn is used to turn a large turbine to produce electricity. This process is similar to many power plants that exist today except that fossil fuel combustion is used as the heat source instead of the sun. The technology is basically environmentally benign, there are no emissions and it does not consume fuel. The only environmental impact is the consumption of land and the release of waste heat. However, the abundant solar resources necessary required for the current technology limit the siting of concentrating thermal to regions like the southwest United States, where the solar resource can be as twice as much as other regions in the United States. Photovoltaics (PV) convert sunlight directly into electricity in a one step process, i.e. photons in and electrons out. When photons

[1] Coal, crude oil (petroleum), natural gas, and propane are all considered fossil fuels because they were formed from the buried remains of plants and animals that lived millions of years ago.

[2] Energy sources are considered nonrenewable if they cannot be replenished (made again) in a short period of time. On the other hand, renewable energy sources such as solar and wind can be replenished naturally in a short period of time. All fossil fuels are nonrenewable but not all nonrenewable fuels are considered fossil fuels. Uranium ore, a solid, is mined and converted to a fuel used at nuclear power plants. Uranium is not a fossil fuel, but is a nonrenewable fuel.

[3] This calculation assumes that about 100 W m^{-2} of solar irradiance hits the earth's surface, 197 million square miles, for 12 hours per day and 365 days per year.

are absorbed in matter, the associated energy is used to excite electrons to higher energy levels. Electrons quickly relax back down to their ground state via phonon or photon emission but in a PV device an asymmetry is built in that causes a permanent electric field to pull the excited electrons away before they are able to relax and feeds them into an external circuit. The extra energy causes a potential difference that is used to drive the electrons through a load in the external circuit. As with concentrating solar thermal technologies, the environmental impact is the consumption of land and the release of waste heat. Commercial PV technologies typically convert light to electricity at an efficiency that ranges between $13.5\% - 17.5\%$ [4].

Coupled with the current state of technology in both solar thermal and PV conversion is the reality that the 'fuel' is intermittent. The term intermittent includes both the concepts of variability and uncertainty. Variability describes the change of generation output due to fluctuations of the irradiance; uncertainty describes the inability to predict in advance the timing and magnitude of the changes in generation output (weather, cloud cover, etc.). This presents a problem for the current state of the electricity infrastructure. The electrical grid is a network made up of electric sources, transmission lines, and electric sinks, which does nothing more than move power from generators to the places of demand via transmission lines and does not store energy. Grid operators must constantly balance power supply with power demand, this maintains the reliability of the grid. Achieving reliability drives up the cost of intermittent source integration, because operators must hold large amounts of reserve capacity[4] to cover an unexpected loss of renewable generation.

Both of the current solar technologies convert the sun's energy to electricity. According to U.S. EIA, electricity is the world's fastest-growing form of end-use energy consumption, as it has been for the past several decades. Net electricity generation worldwide is predicted to rise by 2.3 percent per year on average from 2007 to 2035, while total world energy demand is predicted to grow by 1.4 percent per year [2]. Although this form of end use energy is becoming more prevalent and preferred today, our society still relies heavily on other forms of end use energy. In 2007, electricity made up 13% of end-use energy consumption and is projected to increase its share to 16% by 2035. This means that even if all the electricity consumed was supplied by harnessing the sun's energy, our energy requirements would still need to be supplemented by another energy source or by an advanced yet unknown solar conversion technology.

Realizing that nonrenewable sources of energy are not sustainable, harmful to the environment, and create national security issues *should* be the driving force necessary for humans to change behavior and search for alternatives sources of renewable energy at whatever cost is necessary. We have emphasized should because we live in a society that values energy as a commodity and places a monetary value on it. Having identified the sun as being a potential candidate to replace nonrenewable sources of energy is simply not enough, even if it theoretically can supply all our energy needs. The technological limitations identified significantly harm solar's ability to economically compete with fossil fuel. Levelized Cost of Energy (LCOE) is often used as a metric to compare and assess the overall competitiveness of vastly different electric generating technologies expressed dollars per megawatt-hour (MWh). LCOE reflects the overnight capital cost, fuel cost, fixed and variable O&M cost, financing costs, and an assumed utilization rate for each plant type. In the U.S., it is estimated that the LCOE for some of the more conventional sources of generation are between 65.1 \$/MWh for natural gas combined-cycle plant to 114 \$/MWh for advanced nuclear [5]. However, the LCOE for solar PV is 211 \$/MWh while the LCOE for solar thermal is even less competitive at 312.2 \$/MWh [5].

Increasing the efficiency of solar PV means that a smaller geographic area is required to produce an equivalent amount of electricity. More explicitly, an increase of 50% in energy conversion would reduce the geographic area by 50%. If we assume a linear relationship exists between material costs and efficiency, there would be a reduction in costs of 50% thereby decreasing the LCOE to 105 \$/MWh. This value of LCOE puts solar PV in the same cost range as some of the more conventional electric generating technologies. Is there potential to increase efficiencies of solar PV by this magnitude? Simply put, yes. As mentioned, commercial solar PV modules are only achieving conversion efficiencies about 18%. This would indicate we are still in the beginning stages of perfecting

[4] Utilities have to keep generation capacity on reserve that can be accessed quickly if there is a disruption to the power supply

the technology and significant technology breakthroughs exist.

Although, there are seemingly difficult obstacles in the way of transitioning to renewable energy, as outlined above, solutions exist to each obstacle. Transitioning is not an option but a necessity that will require the inherent human quality of ingenuity that has been demonstrated by past generations. Solutions will not come from any one field of study due to the breadth of the difficulties but rather from multiple fields working together building upon gained knowledge. This thesis is an attempt to contribute one significant solution in the many that are needed. More specifically, it is an attempt at a breakthrough in PV technologies aimed at increasing efficiency based on a high risk, high reward concept that rethinks the solar cell design. Instead of incrementally increasing solar cell efficiency that so much of research today is focused on, e.g. light trapping techniques, advanced anti-reflective coatings, etc., we seek materials that boost efficiencies well beyond the thermodynamic limits restricting conventional cell designs. We look for materials that will utilize the electromagnetic spectrum more efficiently through the introduction of an intermediate band between the conduction and valence band. The aspiration is to create a material breakthrough that will spur additional research on design attempting to mature the technology.

The main purpose of this chapter is to review the design and operating principles of what we term the 'conventional solar cell', that is the commercial cell that is widely distributed today. The review will not be exhaustive but only discuss the necessary fundamentals which the ensuing chapters build upon. In Chap. 2, we introduce, study, and outline the design criteria which is the basis of this thesis, the intermediate band solar cell (IBSC). In Chap. 3, we introduce and study the quantum dot intermediate band solar cell (QD-IBSC) and offer explanations as to why it can be considered a physical realization of the IBSC. In addition, based on certain restrictive assumptions, QD-IBSC material systems are investigated that theoretically increase solar cell efficiencies greater than 70%. In Chap. 4, the finite element method is introduced as a method to solve complicated quantum mechanical problems and we successfully apply the method to Schödinger's equation in one, two, and three dimensions. In Chap. 5, we build upon Chap. 4 to describe how a typical finite element computer code is organized and, in detail, describe the entire finite element program that is used to determine energy levels and corresponding wavefunctions of quantum heterostuctures. In Chap. 6, we will investigate how the strain caused by the growth of quantum dots will induce an additional potential and how to calculate this potential in the framework of the finite element method. We then make use of the code in Chap. 7 to allow us to relax restrictive assumptions and refine our search of QD-IBSC that could theoretically result in solar cell efficiency on the order of $45\% - 70\%$. We conclude in Chap. 8.

1.1 Conventional Photovoltaic Device

When two atoms are brought close together, their orbitals combine and result in levels slightly higher and lower than the original orbitals. The levels split due to the overlapping of wavefunctions. As more atoms are brought closer together, the orbitals split into a large number of levels that are close in energy, effectively forming a continuum or band of levels. The bands formed from different orbitals may or may not overlap and are occupied based on the whether the original orbitals were occupied. The highest occupied band is called the valence band and the lowest unoccupied band is called the conduction band. A metal has overlapping valence and conduction bands, allowing the valence electrons to be easily scattered to nearby energy levels. As such, this allows metals to be good transporters of charge and heat. A semiconductor and insulator[5] have an energy gap E_g with no allowable energy levels between the conduction and valence band (see Fig. 1.1). The electrons in the valence band are completely involved in bonding and will require at least the energy equivalent to the band gap and an unoccupied site in the conduction band in order for the electron to be promoted.

When light is absorbed in matter, electrons can be excited to higher energy levels where they are able to move freely around and, in some cases, emitted from the surface[6] [6]. If the excited electron

[5]The only difference between a semiconductor and insulator is the energy of the band gap. An insulator will have a larger energy gap than a semiconductor such that at room temperature, the conductivity is negligible.

[6]Einstein correctly described this phenomena, the photoelectric effect, in 1905 that later won him the Nobel Price

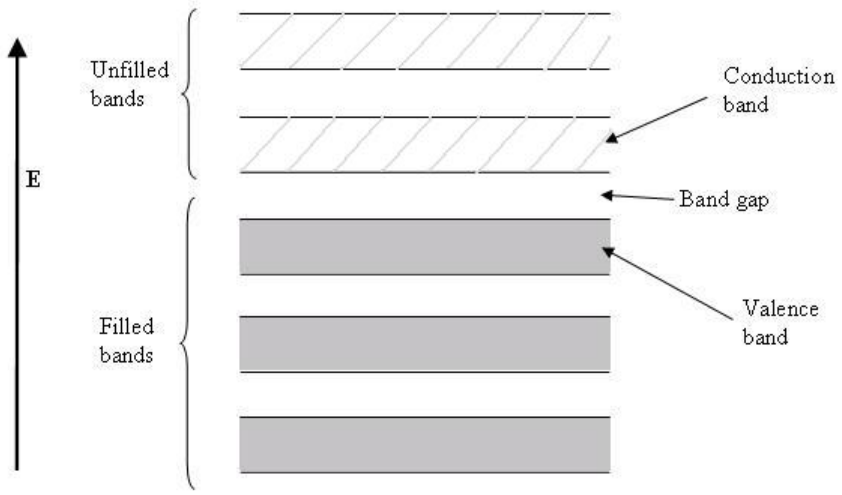


Figure 1.1: A band structure that is representative of a semiconductor or insulator. As many orbitals come together and split, bands are formed. The energy gap between the valence and conduction band is displayed to show there are no allowable energy levels.

is promoted to an energy level within a continuum of energy levels, as in a metal or within the same band, the probability that scattering to a lower energy level occurs will be high and happens on the order of femtoseconds. This is called thermalization and happens through collisions with the lattice, giving up kinetic energy to produce phonons during the decay. However, if an electron is excited across an energy gap to a band containing a continuum of energy levels then the electron will quickly decay to the lowest energy state in the band and then band decay in a much slower process to a vacant site in its previous energy. It can do this via photon emission that is characteristic of the energy gap, E_g. This process is characteristic of a semiconductor or insulator that absorbs a photon of energy greater than the energy gap $\hbar\omega > E_g$ as displayed in Fig. 1.2 and promotes an electron to the relatively empty conduction band. A positively charged vacant site is left behind by the promoted electron that can be filled by another electron. A nearby election can fill the vacancy and the vacancy 'moves' to the neighboring bond. The process can repeat itself and the 'hole' behaves similarly to a positively charged particle. On the other hand, photons of energy less than the energy gap $\hbar\omega < E_g$ do not have the ability to promote an electron to the conduction band.

A photovoltaic device is made up of semiconductor material that takes advantage of the relatively slow band-to-band decay process of electron/hole recombination by having some built-in asymmetry that pulls the electrons away to an external circuit before the electrons can relax back down to the valence band. This is called charge separation and the extra energy creates a potential difference that drives electrons through a load to allow electrical work to be done. It is important to recognize the necessary operating conditions that must be present in order for *any* photovoltaic device to operate. Most obviously, there must be a source of radiation, i.e. the sun. Second, there must be charge generation. By this, we mean that there was an excitation event that has increased the number of free charge carriers, i.e. photon absorption by an electron that has promoted the electron to conduction band and left a hole in the valence band. Lastly, there must be charge separation as described above.

Knowing the operating conditions, we now describe the design of a conventional photovoltaic device. Typically, there are two layers of impurity doped semiconductors, i.e. silicon, placed on a glass substrate. One of the layers is doped with electron donors (n type), the other is doped with electron acceptors (p type). Both of the layers are bound to conducting contacts with external leads,

in 1921.

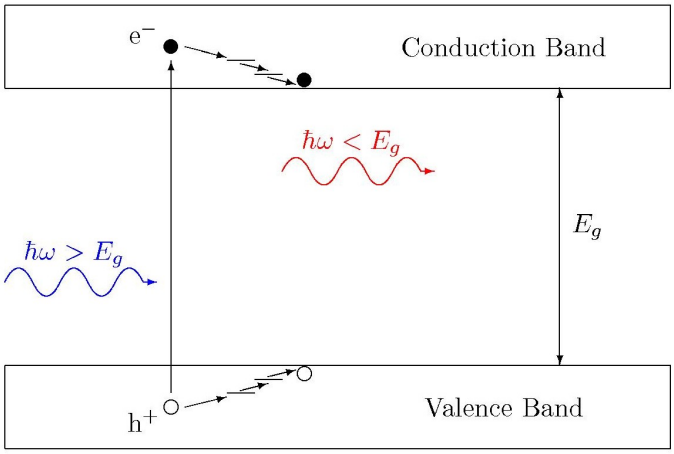


Figure 1.2: A semiconductor with an energy gap of E_g is struck by two photons, a 'red' photon with energy less than E_g and a 'blue' photon with energy greater than E_g. The blue photon is absorbed and promotes an electron to the conduction band creating an electron/hole pair. The electron and hole both thermalize to lowest energy level within the respective band, band edge. In a much slower process, the electron/hole pair will recombine through a photon emission that is characteristic to the energy gap $\hbar\omega = E_g$.

wires, etc. Finally, there is an anti-reflecting coating applied to the top surface with the intent to reduce loss of incident solar radiation by suppressing reflection. In contrast, the glass substrate is typically designed to be reflective so that unabsorbed radiation can pass back through the device. The two impurity doped semiconductors create what is called a p-n junction. Its purpose is to establish a built-in electric field at the interface (junction) between the two layers so that generated electrons flow toward the p side and generated holes flow toward the n side. Hence, its purpose is charge separation.

1.2 Detailed Balance of Efficiency

In the previous section, we briefly described a simple photovoltaic conversion device and the basic operating principles that are needed for operation. By no means was this intended to be exhaustive but rather provide the qualitative understanding necessary for quantitative analysis involving efficiency calculations. In this section, we determine the limiting efficiency of a simple photovoltaic conversion with the view of not just establishing the upper limit but estimating the potential for current design improvement and uncovering the fundamental quantities that determine this upper limit, as these become design criteria for real world devices. In a limited sense this is analogous to Carnots determination of the maximum efficiency that can be attained in the conversion of heat energy to useful work, which provides a quantitative measure of the degree to which the output efficiency of a real heat engine could still be improved. The merit of the heat engine can be appraised in terms of the limit set out by the second law of thermodynamics.

In 1961, Shockley and Queisser published a landmark paper aimed at determining a theoretical justifiable upper efficiency limit [7]. Before then, the treatment of photovoltaic efficiency was based on empirical values for constants describing the characteristics of the conversion device. In general, they were all fairly consistent with observed values and were accepted as theoretical limits. Shockley and Queisser realized that there exists a limit based on a consequence of the nature of atomic processes required by the basic laws of physics and referred to this upper efficiency limit as the detailed balance limit. In determining the detailed balance limit, we define the efficiency η in the usual way as the ratio of power delivered, P_d, to the incident solar power falling on the device, P_s.

For the purposes of this calculation, we model the sun as a blackbody that has a surface temperature of $T_s = 6000$ K. Therefore, the photon and emitted energy flux density respectively are derived from Planck's law over the energy range E_1 and E_2 [8]

$$\dot{N} = \frac{2\pi F(\theta)}{h^3 c^2} \int_{E_1}^{E_2} \frac{E^2}{e^{E/k_B T} - 1} dE \tag{1.1}$$

$$\dot{E} = \frac{2\pi F(\theta)}{h^3 c^2} \int_{E_1}^{E_2} \frac{E^3}{e^{E/k_B T} - 1} dE \tag{1.2}$$

where T is the temperature, k_B is Boltzmann's constant, and $F(\theta) = \sin^2\theta$ is a geometrical factor that arises from integrating over the relevant angular range, where θ, $0 \leq \theta \leq \pi/2$, is defined by the angle to the solar cell surface normal. For the sun as seen from earth, the angle is $\theta = 0.26°$ and $F_{sun}(0.26°) = 2.16 \times 10^{-5}$ [9]. Integrating over all E gives the total emitted power density $F(\theta)\sigma_s T^4$, where σ_s is Stefan's constant.

$$\sigma_s = \frac{2\pi^5 k_B^4}{15 h^3 c^2}$$

At the surface of earth's atmosphere, using the reduced geometric factor F_{sun}, the incoming power density falling on a photovoltaic device with planar geometry is $P_s = 1584$ Watts per meter squared ($\mathrm{W/m^{-2}}$).

Light intensity on a solar cell is called the number of suns, where 1 sun corresponds to standard intensity (1584 $\mathrm{W/m^{-2}}$) or $X = 1$. We are now going to introduce the concept 'concentrated' sunlight[7], which will be used throughout this thesis, as a system or configuration that increases the intensity of light. As such, concentrated sunlight incident on a solar cell would be operating at an increased number of suns. For example $15,840$ $\mathrm{W/m^{-2}}$ incident on a solar cell would be operating at 10 suns, or at $X = 10$. Such practical systems include parabolic reflectors or Fresnel lenses that place the solar conversion device at the focus in either case. Theoretically, a system would achieve the maximum or 'full concentration' when $X = 1/F_{sun} = 46,198$. Incoming power density at full concentration would be the same as the power density at the sun's surface $P_s = \sigma_s T^4 = 63$ $\mathrm{W/m^{-2}}$. Mathematically, increasing concentration is equivalent to increasing the angle for the sun as seen from earth. At this point, it should be recognized that incoming power density P_s varies with the levels of concentration X. After observing that P_s is a dependent variable of device efficiency η, it would be plausible to conclude that η varies with the level concentration of X. We will confirm this premonition in the latter sections of this chapter when device efficiencies with 'unconcentrated' light $X = 1$ are compared to fully concentrated light $X \cdot F_{sun} = 1$.

As a prerequisite to determining the power density delivered to the circuit P_d, it is necessary to discuss electron and hole dynamics in the context of detailed balance. At zero temperature the valence band is filled, while the conduction band is empty. At finite temperatures, some electrons gain enough energy to be excited to the conduction band. The probability that a state with energy E is filled is given by the Fermi-Dirac function [10]

$$f(E,T) = \frac{1}{e^{(E-\mu)/k_B T} + 1} \tag{1.3}$$

where the variables are those as discussed above and μ is the Fermi level. Eq. 1.3 is valid for equilibrium conditions and the probability that an electron is in a state in the conduction or valence band is determined by the single chemical potential. However, when the device is exposed to light or some applied bias, both the electrons and holes are disturbed from their equilibrium. The electron population in the conduction band and hole population in the valence band rise above their normal equilibrium values. This disturbance causes the electrons and holes to relax in what is called quasi-thermal equilibrium. This causes the Fermi energy level to split and electrons in the conduction band settle to a chemical potential μ_n, while the electrons in the valence band settle to a different

[7] This is a standard term that is used in the field and understood as we have defined it in the text.

chemical potential μ_p. They are both assumed to be constant in each band. Using the quasi-thermal equilibrium condition, the probability that a hole is in any state in the valence band is determined by the chemical potential μ_p and the probability that an electron is in any state in the conduction band is determined by μ_n:

$$f_v = \frac{1}{e^{(E_v-\mu_p)/k_BT}+1} \tag{1.4}$$

$$f_c = \frac{1}{e^{(E_c-\mu_n)/k_BT}+1} \tag{1.5}$$

Eq. 1.5 describes the probability distribution for electrons in the valence and conduction band.

In photovoltaic devices, recombination events between the hole and electron degrade efficiency due to the removal of mobile electrons and holes (carriers). Unlike generation, where there is only one mechanism, there are several different recombination mechanisms characterized into two groups: radiative and non-radiative. In radiative recombination, the event produces a photon. Incoming radiation can not only excite an electron to the conduction band creating an electron/hole pair but can also *stimulate* recombination between a mobile electron and hole pair producing an additional photon in the reverse process. It is also well known that excited electrons will *spontaneously* emit a photon in order to recombine with a hole. In non-radiative recombination, the event produces a phonon to release the excess energy. In determining the limiting efficiency, as in the model presented by Shockley and Queisser, any irreversible mechanism is prevented besides those inherent to the photovoltaic operation [11]. Therefore, non-radiative transitions between the conduction and valence bands are assumed absent and radiative recombination sets the upper limit to carrier lifetimes. If radiative recombination is only a fraction of all the recombination, then the efficiency of the device is reduced below the detailed balance limit. Standard reference material for recombination in semiconductors is [12, 13, 14].

Under the assumption that all electronic transitions within a photovoltaic device require photon absorption or emission as required in the detailed balance limit, we can determine the number of recombination events by counting the net number of photons leaving the device. Those electrons will not contribute to the external circuit. Within the device, photons are continuously absorbed and emitted by the processes outlined above but only when one leaves will there be a net electronic transition toward lower energies. Transitions are governed by a first order differential equation using *Fermi's rule* [15] and quasi-thermal probability distribution functions. Let $n(\epsilon, z)$ be the number of photons with energy ϵ that occur at a distance z inside the cell, where we define $z = 0$ at the front surface and $z = 1$ at the back surface. The generation of a photon due to stimulated emission, proportional to $n(\epsilon, z)$, and spontaneous emission, proportional to 1, is [16]

$$\frac{dn(\epsilon,z)}{dz} = \frac{\mathbf{n}}{c}\sum_{i,j} H_{c_i\to v_j}\cdot(n(\epsilon,z)+1)\cdot f_{c_i}\left(1-f_{v_j}\right) \tag{1.6}$$

where $H_{c_i\to v_j}$ is the matrix element coupling the transition from a certain state in the conduction band c_i to a certain state in the valence band v_j, f_{c_i} is the probability that the state c_i is occupied, and $1 - f_{v_j}$ is the probability that the state v_j is not occupied. The sum refers to the different combination of states in the valence and conduction bands producing emission of photons. The factor $\mathbf{n}/c$ ($\mathbf{n}$ is the semiconductor index of refraction, not to be confused with $n(\epsilon, z)$, and c is the speed of light in a vacuum) is intended to transform the time rate into a spacial derivative. Absorption of a photon, proportional to $n(\epsilon, z)$, is

$$\frac{dn(\epsilon,z)}{dz} = -\frac{\mathbf{n}}{c}\sum_{i,j} H_{v_j\to c_i}\cdot n(\epsilon,z)\cdot f_{v_j}\left(1-f_{c_i}\right) \tag{1.7}$$

where symmetry ensures the matrix element $H_{v_j\to c_i} = H_{c_i\to v_j}$. Combining the process of emission

and absorption of photon within the device, we obtain the net photon generation

$$\frac{dn(\epsilon,z)}{dz} = \frac{\mathbf{n}}{c}\left[\sum_{i,j}\left[H_{c_i\to v_j}\cdot(n(\epsilon,z)+1)\cdot f_{c_i}\left(1-f_{v_j}\right) - H_{v_j\to c_i}\cdot n(\epsilon,z)\cdot f_{v_j}\left(1-f_{c_i}\right)\right]\right] \tag{1.8}$$

After some manipulation, Eq. 1.8 is simplified

$$\frac{dn(\epsilon,z)}{dz} = \alpha\cdot(\nu - n(\epsilon,z)) \tag{1.9}$$
$$\alpha = \frac{\mathbf{n}}{c}\sum_{i,j} H_{v_j\to c_i}\cdot\left(f_{v_j}-f_{c_i}\right),\ \nu = \frac{1}{e^{(\epsilon-\mu)k_BT}-1}$$
$$\epsilon = E_{c_i} - E_{v_j},\ \mu = \mu_n - \mu_p$$

Observing that Eq. 1.9 is first order differential equation with integrating factor $e^{\int_0^z \alpha dz}$, the solution is the following:

$$n(\epsilon,z) = n(\epsilon,0)e^{-\alpha z} + \nu(1-e^{-\alpha z}). \tag{1.10}$$

This equation intuitively makes sense, the first part shows the absorption of photons in the mode that have entered from the source. While the latter shows the emission of photons in the mode through the recombination mechanisms described above. As αz becomes large, the exponential is negligible and a constant population independent of αz develops. In the detailed balance limit, the cell is assumed thick enough to achieve full absorption of photons with enough energy to induce radiative transitions between the two bands. For simplicity, we make the assumption that full absorption will be achieved when $z = 1$ and the emitted photon population from the device in each mode is in thermal equilibrium

$$n(\epsilon,1) = \left\{ \begin{array}{ll} 0, & 0\le\epsilon<E_g \\ \frac{1}{e^{(\epsilon-\mu)/k_BT}-1}, & \epsilon\ge E_g \end{array} \right\} \tag{1.11}$$

with a band gap of E_g. For $\epsilon \ge E_g$, Eq. 1.11 gives the Bose-Einstein mean occupation number of photons with energy ϵ [17]. Thus, the photon flux density behaves like a blackbody flux density as in Eq. 1.1 and we generalize to define the function

$$N(E_1,E_2,T,\mu) = \frac{2\pi}{h^3c^2}\int_{E_1}^{E_2}\frac{E^2}{e^{(E-\mu)/k_BT}-1}dE \tag{1.12}$$

where all the variables are those outlined above. Equation 1.12 does not include the geometrical factor $F(\theta) = \sin^2\theta$ and as a result, the photon flux density is $F(\theta)\cdot N(E_1,E_2,T,\mu)$ to account for the relevant angle range. As an example, lets consider the radiation leaving the solar conversion device that has a perfect mirror located on the back of the cell so that radiation makes a double pass through the cell and can only escape through the front area of illumination, i.e. $\theta = \pi/2$ see Eg. 1.1-1.2. The photon flux density would be described using Eqs. 1.11-1.12 and $F(\pi/2)$ as $N(E_g,\infty,T_a,\mu)$, where T_a is the solar device's characteristic temperature.

We are now going to compute the power density P_d delivered by a photovoltaic device based on what we have developed thus far. This might seem odd at a first considering that we have only discussed incoming and outgoing photon fluxes but it will become clear as certain assumptions about the conversion device and incoming radiation are described (some of which have already been discussed). Let us first begin by reviewing the efficiency equation of the photovoltaic conversion device

$$\eta = \frac{A_c\cdot P_d}{A_c\cdot P_s} \to \frac{J_m\cdot V_m}{X\cdot F_{sun}\sigma_s T_s^4} \to \frac{J_m\cdot V_m}{X\cdot 1584} \tag{1.13}$$

where A_c is the area of the conversion device, P_d is the maximum power density delivered by the device to an external circuit expressed as the current density J_m multiplied by the voltage V_m, and P_s is the power density received by the device from the sun. We have made use of the concentration

factor X, geometric factor $F_{sun} = 2.16 \times 10^{-5}$, Stefan's constant σ_s, and temperature of the sun $T_s = 6000$ K. The following is a list of assumptions that are used in determining the detailed balance efficiency [7, 18]:

1. The solar cell absorbs blackbody radiation at a temperature of T_s =6000°K and ambient T_a =300°K and emits blackbody radiation at ambient T_a =300°K;
2. Only radiative transitions occur between the bands;
3. All photons above the lowest energy gap are absorbed;
4. Carrier mobility is infinite and as a consequence, the quasi-Fermi energy levels are constant throughout the cell so $\mu = qV$;
5. Only one electron-hole pair is created per photon;
6. A perfect mirror is located on the back of the device so that radiation makes a double pass through the cell and can only escape through the front area of illumination;
7. The net photon flux (number of incident minus number of emitted photons) is equal to the number of charge carrier pairs collected at the contacts (detailed balance assumption).

It is important to understand that as photons with energy greater than the bandgap are absorbed to create an electron-hole pair, any excess energy beyond that of the bandgap will be lost due to thermalization and the carriers will relax to the band edges before circuit extraction or recombination. This makes an absorbed photon with $\hbar\omega > E_g$ have the same effect as an absorbed photon with energy $\hbar\omega = E_g$. It is this reason why we are concerned with the photon flux density and not the photon energy density.

Since only radiative events are considered, generation and recombination events are signaled by photon absorption or emission. In addition, we make the assumption that the device has perfect charge carrier collection meaning that photogenerated charge carriers surviving radiative recombination will be collected by the external circuit. Therefore, the net photon flux will be equal to the number of charge carrier pairs collected at the contacts and be equal to charge carrier flux. When the charge carrier flux is multiplied by the electric charge q, the current density of the device is found

$$\begin{aligned} J(E_g, T_s, T_a, X, V) &= q[XF_{sun}N(E_g, \infty, T_s, 0) + (1 - XF_{sun})N(E_g, \infty, T_a, 0) \\ &- F(\pi/2)N(E_g, \infty, T_a, qV)] \end{aligned} \tag{1.14}$$

where we have made use of Eq. 1.12, the geometric factor $F(\theta)$, and the concentration factor X. The first term on the right hand side in the equation represents the current density generated from radiation the cell absorbs from the sun at the characteristic temperature T_s over the angular range $0 < \theta < 0.26°$, the second term in the equation represents the current density generated from radiation the cell absorbs from ambient at the characteristic temperature T_a over the angular range $0.26° < \theta < \pi/2$[8], and the last term in the equation represents the recombination of carriers through photon emission at the characteristic temperature T_a and uniform potential qV that do not contribute to the current density over the angular range $0 < \theta < \pi/2$.

For each value of E_g, there exists a voltage V_m that maximizes the power density delivered by the device such that maximum efficiency of the device is

$$\eta(E_g, T_s, T_a, X) = \frac{J(E_g, T_s, T_a, X, V_m) \cdot V_m}{X \cdot 1584}. \tag{1.15}$$

Following the prescription of finding the detailed balance limiting efficiency as outlined above, we have found the maximum efficiency as a function of the band gap E_g for unconcentrated light

[8] Photon absorption includes a contribution from thermal photons that is assumed to behave like a blackbody over the rest of the hemisphere. Although the contribution to the current is negligible, it is a standard assumption in efficiency calculations, see [16, 9, 19].

$\eta(E_g, T_s = 6000^o K, T_a = 300^o K, X = 1)$ and fully concentrated light $\eta(E_g, T_s = 6000^o K, T_a = 300^o K, X = 1/F_{sun})$ (see Fig. 1.3). Fully concentrated light has a maximum efficiency of 41% when

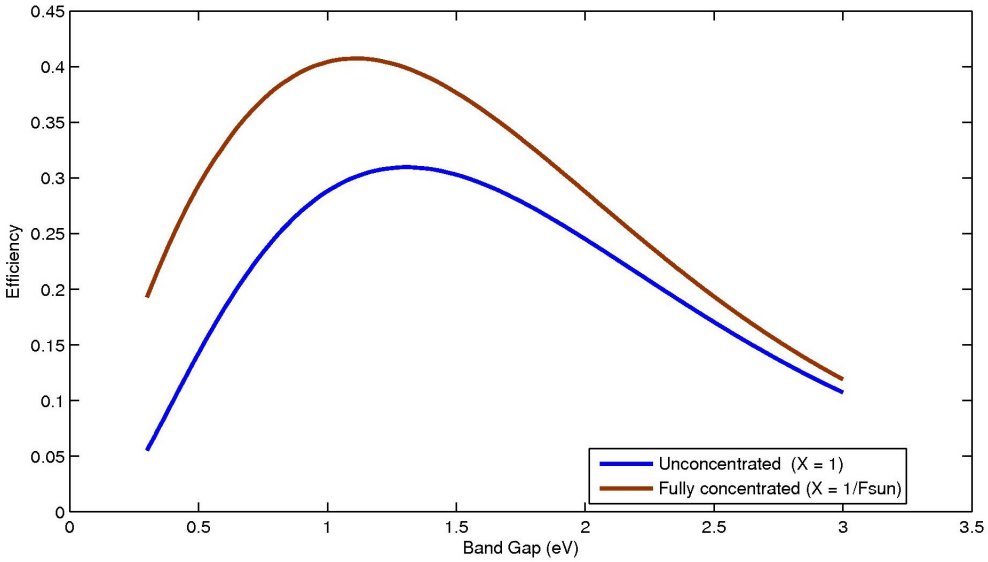


Figure 1.3: Detail balance limiting efficiency of a photovoltaic device as a function of the band gap for unconcentrated light $X = 1$ and fully concentrated light $X = 1/F_{sun}$. The efficiency was calculated assuming the sun is modeled as a blackbody with characteristic temperature $6000^o K$. For fully concentrated light, the maximum efficiency of about 41% occurs approximately at a band gap of $E_g = 1.1$ eV, while unconcentrated light leads to a maximum efficiency of about 31% with a band gap of $E_g = 1.31$ eV.

there is a band gap of 1.1 eV, while unconcentrated light has a maximum efficiency of 31% when there is a band gap of 1.31 eV. It is clear the assumption that limiting efficiency does change with different concentration factors is correct. More specifically, as the concentration factor increases so does the limiting efficiency up to the maximum of 41%.

Efficiency of the single energy gap solar conversion device seems to drop off for small energy gaps, $E_g < 1$ eV, and for larger energy gaps, $E_g > 2$ eV. Intuitively this can be understand in terms of the power density $P_d = J \cdot V$ that is delivered by the device. For smaller band gaps, most photons would be absorbed thereby increasing current density but a good portion of the their energy would be wasted through thermalization. In addition, since the cell operates at a potential difference proportional to the band gap, each extracted carrier's potential energy will be small. For larger band gaps, a good portion of the incident photons would not be absorbed and current density would decrease. In both cases, efficiency drops off due to the current density being inversely proportional to the operating voltage.

From the detail balance limiting efficiency calculations, we conclude that efficiency is highly dependent on the device's band gap and that photons with energy near the band gap are used most efficiency. If the solar resource were monochromatic, conversion would be optimal because the device's band gap would be tuned to match the light's energy. Unfortunately the solar resource is not monochromatic and with a single band gap, low energy photons are not absorbed and higher energy photons lose energy through thermalization. As such the best conversion based on a single band gap is 41%.

We now turn our focus on ways to better utilize our solar resource's radiation.

Chapter 2: Intermediate Band Solar Cell

In Chap. 1, we saw how to calculate the limiting efficiency of the conventional solar cell through a detailed balance argument that follows from a series of assumptions. We found that when the conversion device absorbs incident unconcentrated light efficiency was 31% and could be increased to 41% when the conversion device absorbs fully concentrated light. These two limits set the lower and upper bound for the limiting efficiency of any concentrating light system. We concluded that efficiency is highly dependent on the device's band gap and that photons with energy near the band gap are used most efficiency. The work done per photon decreases as the photon's energy increases beyond the band gap with losses occurring from thermalization and goes to zero when the photon's energy is less than the band gap. These losses are realized in the conventional solar cell because our solar resource has a broad energy spectrum and poorly matches the band gap, as reflected in the limiting efficiency values.

The main limitations of the photovoltaic conversion device is that low energy photons cannot excite charge carriers to the conduction band, therefore do not contribute to the devices's current, and high energy photons are not efficiently used due to a poor match to the energy gap. However, if intermediate levels are introduced into the energy gap of a conventional solar cell, then low energy photons can be used to promote charge carriers in a stepwise manner to the conduction band. In addition, the photons would be better matched with energy transitions between bands. Fig. 2.1 illustrates this type of structure. In this case, there are two intermediate bands between the valence and conduction bands, allowing for a total of six upward electronic transitions. This type of device is called an intermediate band solar cell (IBSC) [16]. This is a multi-step or ladder approach to increase efficiency. It will shown that the maximum efficiency of a photovoltaic conversion device using one or two intermediate bands is greater than the single band gap conventional device.

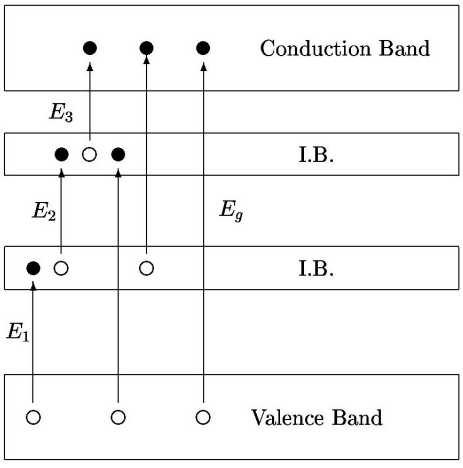


Figure 2.1: This 4-Band diagram depicts two intermediate bands and shows the possible electronic transitions (reverse transitions are not shown but do occur). There is a total of six upward transitions, with E_1, E_2, and E_3 making up the three independent ones. The large energy gap, E_g, is the normal gap between the conduction and valence bands.

The objective of this chapter is to understand the IBSC concept and perform some useful efficiency calculations. We begin in Sec. 2.1 with a description of how the IBSC is presumed to operate and perform a detailed balance analysis on a photovoltaic conversion device with one intermediate

band. In addition, the sensitivity of the efficiency as a function of the intermediate band energy level is investigated. The next section, Sec. 2.2, continues to develop the IBSC concept in a logical manner by considering two intermediate bands. We conclude with Sec. 2.3.

2.1 One Intermediate Band Detailed Balance Analysis

In a device where there is one intermediate band located between the conduction and valence band, an electron from the valence band can be excited to either the intermediate or conduction band. Additionally, an electron located in the intermediate band can be excited to the conduction band. In total, there are three upward energy transitions in this device: E_1, E_2, and E_g. E_1 represents valence to intermediate band, E_2 represents intermediate to conduction band, and E_g represents the conventional band gap between the valence and conduction band. The two intermediate transitions E_1 and E_2 are independent of each other, while the band gap transition E_g is a function of the two intermediate ones: $E_g = E_1 + E_2$. Proper operation of the IBSC requires that no charge carriers are to be extracted from the intermediate band [16]. This is important for two reasons. First, this would deplete the carrier density available for excitation to higher bands, including the conduction band. Second, charge carriers are extracted from intermediate bands at lower potential differences than from the conduction band. This would reduce the power output of the IBSC and thus the conversion efficiency. Since carriers would only be extracted from the valence and conduction bands, the IBSC would operate similar to the conventional solar cell in the sense that the operating voltage would be proportional to E_g.

When the conventional solar cell is exposed to light or some applied bias, carrier populations increase beyond equilibrium and cause the Fermi energy level to split into two chemical potentials μ_n and μ_p. These chemical potentials represent steady state solutions that allow treatment of the carrier dynamics in each band separately. More specifically, they are used to determine the probability that an electron will be in any state in the conduction band and a hole will be in any state in the valence band (see Eq. 1.5). When the IBSC is exposed to light or some applied bias, the Fermi energy level must split into the number of total bands. In the case with one intermediate band, the Fermi energy level will split into three chemical potentials μ_n, μ_i, and μ_p that represent steady state solutions that allow treatment of the carrier dynamics in the conduction, intermediate, and valence band respectively.

$$f_v = \frac{1}{e^{(E_v-\mu_p)/k_BT}+1} \tag{2.1}$$

$$f_c = \frac{1}{e^{(E_c-\mu_n)/k_BT}+1} \tag{2.2}$$

$$f_I = \frac{1}{e^{(E_I-\mu_i)/k_BT}+1} \tag{2.3}$$

Each band in the IBSC must possess its own chemical potential. If the intermediate band is thermally coupled to the conduction or valence band, excited electrons will lose kinetic energy through phonon emission before the necessary next event occurs for proper IBSC operation: (1) electrons are not extracted from conduction band or (2) electrons are not promoted from the intermediate to conduction band. As a result, the Fermi energy level will split into two chemical potentials and carrier dynamics will be dominated by these two chemical potentials as in the conventional solar conversion device.

Using performance characteristics and assumptions of the IBSC, we can perform limiting efficiency calculations similar to the detailed balance calculation developed by Shockley and Queisser. As with the conventional solar conversion device in determining the limiting efficiency, only radiative recombination exists so that all electronic transitions within a photovoltaic device require photon absorption or emission. Therefore, we can determine the number of recombination events by counting the net number of photons leaving the device. This sets up a differential equation that is a bit more complicated than Eq. 1.8 using *Fermi's rule* and the quasi-thermal probability distribution functions. We let $n(\epsilon, z)$ be the number of photons with energy ϵ that occur at a distance z inside the

cell, where we define $z = 0$ at the front surface and $z = 1$ at the back surface. Unlike the convention solar cell, the IBSC has three upward and three downward energy transitions with matrix elements coupling each transition. Due to symmetry, we make the assumption that the upward matrix element coupling a certain transition will be equal to the downward matrix element coupling the same transition. We let the matrix element $H_{v_j \to c_i} = H_{c_i \to v_j}$ couple a certain state in the conduction band c_i to a certain state in the valence band v_j. We let the matrix element $H_{v_j \to I_i} = H_{I_i \to v_j}$ couple a certain state in the intermediate band I_i to a certain state in the valence band v_j. We let the matrix element $H_{c_j \to I_i} = H_{I_i \to c_j}$ couple a certain state in the conduction band c_j to a certain state in the intermediate band I_i. Combining the process of emission and absorption of photon within the device, we obtain the net photon generation for the IBSC

$$\begin{aligned}
\frac{dn(\epsilon,z)}{dz} &= \frac{\mathbf{n}}{c}\sum_{i,j} H_{c_i \to v_j} \cdot (n(\epsilon,z)+1) \cdot f_{c_i}\left(1-f_{v_j}\right) - H_{v_j \to c_i} \cdot n(\epsilon,z) \cdot f_{v_j}\left(1-f_{c_i}\right) \\
&+ \frac{\mathbf{n}}{c}\sum_{i,j} H_{c_i \to I_j} \cdot (n(\epsilon,z)+1) \cdot f_{c_i}\left(1-f_{I_j}\right) - H_{I_j \to c_i} \cdot n(\epsilon,z) \cdot f_{I_j}\left(1-f_{c_i}\right) \\
&+ \frac{\mathbf{n}}{c}\sum_{i,j} H_{I_i \to v_j} \cdot (n(\epsilon,z)+1) \cdot f_{I_i}\left(1-f_{v_j}\right) - H_{v_j \to I_i} \cdot n(\epsilon,z) \cdot f_{v_j}\left(1-f_{I_i}\right)
\end{aligned} \tag{2.4}$$

where each line represents one of the three transitions. The sum refers to the different combination of states in the two bands producing emission of photons. The factor $\mathbf{n}/c$ ($\mathbf{n}$ is the semiconductor index of refraction,ot to be confused with $n(\epsilon,z)$, and c is the speed of light in a vacuum) is intended to transform the time rate into a spacial derivative.

After some manipulation, Eq. 2.5 is simplified

$$\begin{aligned}
\frac{dn(\epsilon,z)}{dz} &= \alpha_{cv}\cdot(\nu_{cv}-n(\epsilon,z)) + \alpha_{cI}\cdot(\nu_{cI}-n(\epsilon,z)) + \alpha_{Iv}\cdot(\nu_{Iv}-n(\epsilon,z)) \\
\alpha_{cv} &= \frac{\mathbf{n}}{c}\sum_{i,j} H_{v_j \to c_i}\cdot\left(f_{c_j}-f_{v_i}\right),\ \nu_{cv} = \frac{1}{e^{(\epsilon-\mu_{cv})k_BT}-1} \\
\alpha_{cI} &= \frac{\mathbf{n}}{c}\sum_{i,j} H_{I_j \to c_i}\cdot\left(f_{c_j}-f_{I_i}\right),\ \nu_{cI} = \frac{1}{e^{(\epsilon-\mu_{cI})k_BT}-1} \\
\alpha_{Iv} &= \frac{\mathbf{n}}{c}\sum_{i,j} H_{v_j \to I_i}\cdot\left(f_{I_j}-f_{v_i}\right),\ \nu_{Iv} = \frac{1}{e^{(\epsilon-\mu_{Iv})k_BT}-1} \\
\epsilon &= E_{c_j}-E_{v_i} = E_{c_j}-E_{I_i} = E_{I_j}-E_{v_i} \\
\mu_{cv} &= \mu_n-\mu_p,\ \mu_{cI} = \mu_n-\mu_I,\ \mu_{Iv} = \mu_I-\mu_p
\end{aligned} \tag{2.5}$$

Observing that Eq. 2.5 is first order differential equation with integrating factor

$$e^{\int_0^z(\alpha_{cv}+\alpha_{cI}+\alpha_{Iv})dz},$$

the solution is the following:

$$n(\epsilon,z) = n(\epsilon,0)e^{-(\alpha_{cv}+\alpha_{cI}+\alpha_{Iv})z} + \frac{\alpha_{cv}\nu_{cv}+\alpha_{cI}\nu_{cI}+\alpha_{Iv}\nu_{Iv}}{\alpha_{cv}+\alpha_{cI}+\alpha_{Iv}}\cdot\left(1-e^{-(\alpha_{cv}+\alpha_{cI}+\alpha_{Iv})z}\right) \tag{2.6}$$

Similar to the Eq. 1.10, the first part of the expression on the right shows the absorption of photons in the mode that have entered from the source. While the second part of the expression on the right shows the emission of photons in the mode through the radiative recombination mechanisms. As the term $(\alpha_{cv}+\alpha_{cI}+\alpha_{Iv})z$ becomes large, the exponential is negligible and a constant population independent of $(\alpha_{cv}+\alpha_{cI}+\alpha_{Iv})z$ develops in the form of the average of the three Bose-Einstein functions weighted by the absorption coefficients: α_{cv}, α_{cI}, and α_{Iv}. In the limiting efficiency, the IBSC is assumed thick enough to achieve full absorption of photons to induce radiative transitions between two bands for the relevant range of energies such that only one Bose-Einstein function

is needed to describe the photon population. For simplicity, we make the assumption that full absorption will be achieved when $z = 1$ and the emitted photon population from the device in each mode is in thermal equilibrium

$$n(\epsilon, 1) = \left\{ \begin{array}{ll} 0, & 0 \leq \epsilon < E_1 \\ \frac{1}{e^{(\epsilon - \mu_{Iv})/k_B T} - 1}, & E_1 \leq \epsilon < E_2 \\ \frac{1}{e^{(\epsilon - \mu_{cI})/k_B T} - 1}, & E_2 \leq \epsilon < E_g \\ \frac{1}{e^{(\epsilon - \mu_{cv})/k_B T} - 1}, & \epsilon \geq E_g \end{array} \right\} \tag{2.7}$$

with a band gap of $E_g = E_1 + E_2$, E_1 represents the energy gap between the valence band and intermediate band, and E_2represents the energy gap between the intermediate band and conduction band. For each energy range $\epsilon \geq E_1$, Eq. 2.7 is the Bose-Einstein mean occupation number of photons with energy ϵ [17]. Thus, the photon flux density behaves like a blackbody flux density as in Eq. 1.1 and as we did in Chap. 1.2 we generalize to define the function

$$N(E_1, E_2, T, \mu) = \frac{2\pi}{h^3 c^2} \int_{E_1}^{E_2} \frac{E^2}{e^{(E-\mu)/k_B T} - 1} dE \tag{2.8}$$

As before, Eq. 2.8 does not include the geometrical factor $F(\theta) = \sin^2 \theta$ and as a result, the photon flux density is $F(\theta) \cdot N(E_1, E_2, T, \mu)$ to account for the relevant angle range.

We are now going to develop the limiting efficiency of the IBSC with one intermediate band using arguments similar to the detailed balance efficiency. The following list of assumptions is used to carry out limiting efficiency calculations for the IBSC [7, 16]:

1. The solar cell absorbs blackbody radiation at a temperature of T_s =6000°K and ambient T_a =300°K and emits blackbody radiation at ambient T_a =300°K;
2. Only radiative transitions occur between the bands;
3. All photons above the lowest energy gap are absorbed and no high energy photon is used in a low energy process;
4. Carrier mobility is infinite and as a consequence, the quasi-Fermi energy levels are constant throughout the cell;
5. Only one electron-hole pair is created per photon;
6. A perfect mirror is located on the back of the cell so that radiation makes a double pass through the cell and can only escape through the front area of illumination;
7. No carriers are extracted from the intermediate band(s);
8. The net photon flux (number of incident minus number of emitted photons) is equal to the number of charge carrier pairs collected at the contacts (detailed balance assumption).

The efficiency equation given by Eq. 1.13 is worth repeating

$$\eta = \frac{J_m \cdot V_m}{X \cdot 1584} \tag{2.9}$$

where the power density delivered by the device is current density J_m multiplied by the voltage V_m. To find the limiting efficiency, our task lies in finding the characteristic IBSC parameters that maximize the delivered power density.

Photons with energy $E_1 \leq \hbar\omega < E_2$ are absorbed to promote an electron to the intermediate band and create a hole in the valence band, any excess energy beyond that of the energy transition E_1 will be lost due to thermalization and carriers will relax to the band edges before another radiative event occurs. This makes an absorbed photon with $E_1 < \hbar\omega < E_2$ have the same effect as an absorbed photon with energy $\hbar\omega = E_1$. Photons with energy $E_2 \leq \hbar\omega < E_g$ are absorbed to promote an

electron from the intermediate band to the conduction band, any excess energy beyond that of the energy transition E_2 will be lost due to thermalization and carriers will relax to the band edges before circuit extraction or recombination. This makes an absorbed photon with $E_2 < \hbar\omega < E_g$ have the same effect as an absorbed photon with energy $\hbar\omega = E_2$. Photons with energy $\hbar\omega \geq E_g$ are absorbed to promote an electron to the conduction band and create a hole in the valence band, any excess energy beyond that of the energy transition E_g will be lost due to thermalization and carriers will relax to the band edges before before circuit extraction or recombination. This makes an absorbed photon with $\hbar\omega > E_g$ have the same effect as an absorbed photon with energy $\hbar\omega = E_g$. The net photon flux is equal to the number of charge carrier pairs collected at the contacts so that the current density is just

$$\begin{aligned} &J(E_1, E_2, E_g, T_s, T_a, X, V) = \\ &q[XF_{sun}N(E_g, \infty, T_s, 0) + (1 - XF_{sun})N(E_g, \infty, T_a, 0) - F(\pi/2)N(E_g, \infty, T_a, qV)] \\ &+q[XF_{sun}N(E_2, E_g, T_s, 0) + (1 - XF_{sun})N(E_2, E_g, T_a, 0) - F(\pi/2)N(E_2, E_q, T_a, \mu_{cI})] \end{aligned} \tag{2.10}$$

where we have made use of Eq. 1.12, the geometric factor $F(\theta)$, and the concentration factor X. The terms in the first bracket on the right hand side of the equation represent the current density generated from the promotion of electrons from the valence band to the conduction band less recombination events from the reverse transition, while the terms in the second bracket represent the current density generated from the promotion of electrons from the intermediate band to the conduction band less recombination events from the reverse transition. In both bracketed terms, the IBSC absorbs radiation from the sun at the characteristic temperature T_s over the angular range $0 < \theta < 0.26°$ and from ambient at the characteristic temperature T_a over the angular range $0.26° < \theta < \pi/2$, while the IBSC emits radiation at the characteristic temperature T_a and characteristic uniform chemical potential over the angular range $0 < \theta < \pi/2$.

For each energy configuration (E_1, E_2, E_g), there exists a voltage V_m that maximizes the IBSC power density output $V_m \cdot J_m(E_1, E_2, E_g, T_s, T_a, X, V)$. However, the current density equation contains an additional chemical potential μ_{cI} and we must assign a value if we are going to determine Eq. 2.10. Proper operation of the IBSC requires that there is no current extracted from the intermediate band(s), i.e. the current entering the intermediate must equal the current leaving the intermediate band. This sets up the constraint

$$\begin{aligned} &q[XF_{sun}N(E_2, E_g, T_s, 0) + (1 - XF_{sun})N(E_2, E_g, T_a, 0) - F(\pi/2)N(E_2, E_q, T_a, \mu_{cI})] \\ &= q[XF_{sun}N(E_1, E_2, T_s, 0) + (1 - XF_{sun})N(E_1, E_2, T_a, 0) - F(\pi/2)N(E_1, E_2, T_a, \mu_{Iv})] \end{aligned} \tag{2.11}$$

and when considered in conjunction with

$$qV = \mu_{cv} = \mu_{cI} + \mu_{Iv} \tag{2.12}$$

for a given value V, all the chemical potentials are determined and the current density for the IBSC is solved.

Following the prescription outlined above, we have found the limiting efficiency of the IBSC as a function of the band gap E_g by varying the energy transition E_1 above the valence band for unconcentrated $X = 1$ and fully concentrated $X = 1/F_{sun}$ light (see Fig. 2.2). Fully concentrated light has a maximum efficiency of 63.2% when the largest band gap is $E_g = 1.93$ eV, while unconcentrated light has a maximum efficiency of 46.8% when the largest band gap is $E_g = 2.40$ eV. As the concentration factor increases, so does the limiting efficiency up to the maximum of 63.2%, a result that is similar to the conventional solar conversion device. We should note that two patterns have emerged when considering limiting efficiency calculations for both the IBSC and conventional solar conversion device: (1) as light concentration increases, efficiency increases and (2) the large band gap E_g decreases as light concentration increases.

As we compare the performance of the one intermediate band IBSC model to the conventional solar conversion model, it is evident that conversion efficiency has significantly improved. For unconcentrated light, performance has increased from 31% to 46.8% which is a 33.7% increase in efficiency.

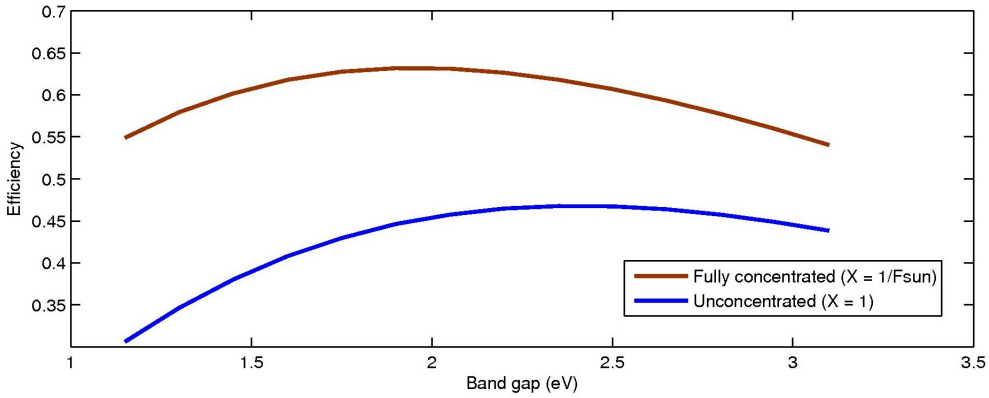


Figure 2.2: Limiting efficiency of the IBSC with one intermediate band as a function of the largest band gap E_g for unconcentrated light $X = 1$ and fully concentrated light $X = 1/F_{sun}$. The efficiency was calculated assuming the sun is modeled as a blackbody with characteristic temperature $6000^{o}K$. For fully concentrated light, the maximum efficiency of about 63.2% occurs approximately at a band gap of $E_g = 1.93$ eV, while unconcentrated light leads to a maximum efficiency of about 46.8% with a band gap of $E_g = 2.40$ eV.

For fully concentrated light, performance has increased from 41% to 63.2% which is a 35% increase in efficiency. This increase is dramatic and the motivating factor for further research in this type of structure.

After calculating the significant efficiency improvements of the IBSC, the interest surrounding the sensitivity of efficiency as a function of the two independent energy transitions E_1 and E_2 should be investigated. Presumingly, a physical device that employs the characteristics of an IBSC will not exactly match the energy band configuration that leads to the theoretical maximum efficiency. Therefore, it would be prudent to calculate the sensitivity of these energy levels to see how efficiency responds. Does the limiting efficiency drop off suddenly if energy levels are slightly removed from the optimized position? To answer this question, we calculated the limiting efficiency of the IBSC as a function of the two energy transitions E_1 and E_2 for both unconcentrated and fully concentrated light (see Figs. 2.3 and 2.4).

For unconcentrated light, the maximum efficiency of 46.8% occurs at the energy band transitions $E_1 = 0.92$ eV and $E_2 = 1.48$ eV but Fig. 2.3 shows that an efficiency of $\geq 46\%$ can occur at various band configurations. In comparison with the conventional conversion device, most energy band configurations of the IBSC have a greater limiting efficiency than 31%. Additionally, it does not seem that efficiency drops off suddenly for band configurations outside the optimized position.

From Fig. 2.3, the IBSC can achieve a limiting efficiency $\geq 46\%$ for band configurations ranging between 0.76 eV $\leq E_1 \leq$ 1.06 eV and 1.30 eV $\leq E_2 \leq$ 1.65 eV. A useful quantitative computation is to find analytic expressions for the interval of $E_2 = [a, b]$ that results in an efficiency $\geq 46\%$ given that E_1 falls in the interval of $[0.76, 1.06]$. To do this, we fit a cubic polynomial to describe the endpoints a and b of interval $[a, b]$. We choose a cubic polynomial rather than higher degree polynomials because it is the lowest degree that accurately describes the endpoints.

$$\begin{aligned} a(E_1) &= 0.5689 \cdot (E_1)^3 - 0.5638 \cdot (E_1)^2 + 0.6593 \cdot E_1 + 0.8796; \quad 0.76 \leq E_1 \leq 1.06 \\ b(E_1) &= 11.760 \cdot (E_1)^3 - 34.351 \cdot (E_1)^2 + 34.119 \cdot E_1 - 9.9329; \quad 0.76 \leq E_1 \leq 1.06 \end{aligned} \tag{2.13}$$

The two expressions show that for a given E_1 within the interval $[0.76, 1.06]$, the efficiency of the IBSC will be greater than or equal to 46% when E_2 is within the range $[a, b]$. The interval $[a, b]$ can further be utilized in the design process of an IBSC for unconcentrated light by selecting a device that displays modeling results of 0.76 eV $\leq E_1 \leq$ 1.06 and $E_2 = (b(E_1 + a(E_1)))/2$ to allow for maximum uncertainty in the physical realization of the device.

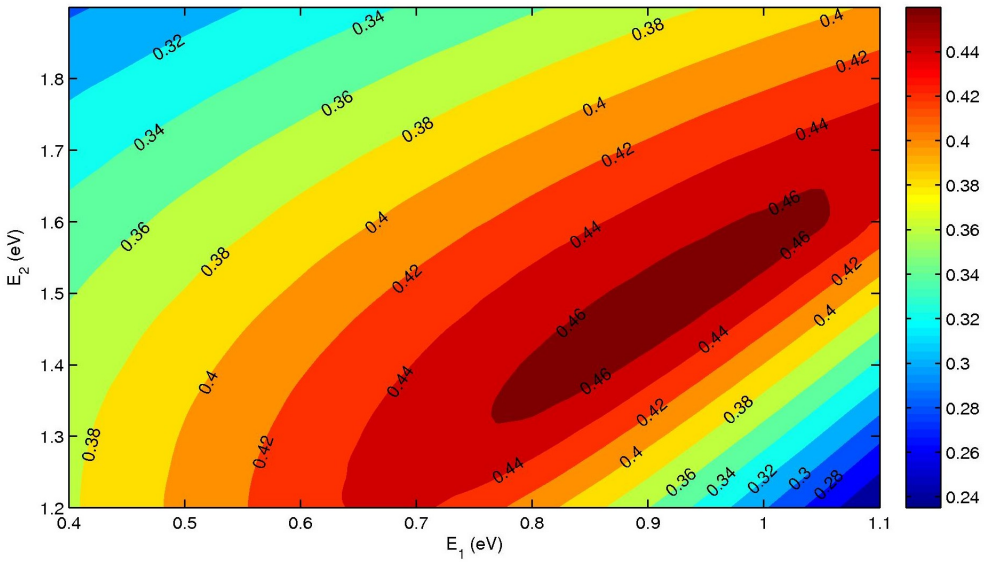


Figure 2.3: Limiting efficiency of the IBSC with one intermediate band as a function of the two energy transitions E_1 and E_2 for unconcentrated light $X = 1$. The color bar on the right hand side of the contour plot summarizes what efficiencies the various colors represent.

For fully concentrated light, the maximum efficiency of 63.2% occurs at the energy band transitions $E_1 = 0.70$ eV and $E_2 = 1.23$ eV but Fig. 2.4 shows that an efficiency of $\geq 62\%$ can occur at various band configurations. Specifically at band configurations ranging in the intervals 0.57 eV $\leq E_1 \leq 0.88$ eV and 1.06 eV $\leq E_2 \leq 1.43$ eV. In comparison with the conventional conversion device, most energy band configurations of the IBSC have a greater limiting efficiency than 41%. Again, it does not seem that efficiency drops off suddenly for band configurations outside the optimized position.

We perform a similar computation to find analytic expressions for the interval of $E_2 = [a, b]$ that results in an efficiency $\geq 62\%$ given that 0.57 eV $\leq E_1 \leq 0.88$. A cubic polynomial function is fit to describe the endpoints a and b. Again, we choose a cubic polynomial rather than higher degree polynomials because it is the lowest degree that accurately describes the endpoints

$$\begin{aligned} a(E_1) &= -0.8724 \cdot (E_1)^3 + 2.7654 \cdot (E_1)^2 - 1.4265 \cdot E_1 + 1.1338; \;\; 0.57 \leq E_1 \leq 0.88 \\ b(E_1) &= 5.2569 \cdot (E_1)^3 - 13.536 \cdot (E_1)^2 + 12.355 \cdot E_1 - 2.5355; \;\; 0.57 \leq E_1 \leq 0.88 \end{aligned} \quad (2.14)$$

The two expressions show that for a given E_1 within the interval $[0.57, 0.88]$, the efficiency of the IBSC will be greater than or equal to 62% when E_2 is within the range $[a, b]$. The interval $[a, b]$ can further be utilized in the design process of an IBSC for fully concentrated light by selecting a device that displays modeling results of 0.57 eV $\leq E_1 \leq 0.88$ and $E_2 = (b(E_1) + a(E_1))/2$ to allow for maximum uncertainty in the physical realization of the device.

Before we proceed to an analysis of the limiting efficiency of the IBSC containing two intermediate bands, it is important to discuss the integral from Eq. 2.8 and method for solving the chemical potentials from a practical standpoint. The flux integral plays an important role in solving for the limiting efficiency of solar conversion devices as we have outlined in the previous sections. It is part of a class of integrals called Bose-Einstein integrals that are seen throughout physics. Therefore much effort has been undertaken in studying the properties and solutions, both analytical and numerical. For the purposes of the work here, only numerical solutions are of interest due to the relative ease

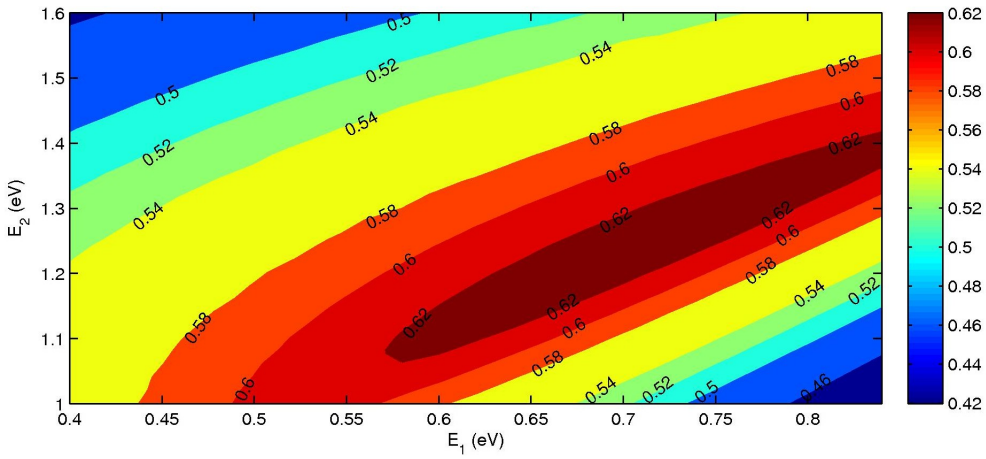


Figure 2.4: Limiting efficiency of the IBSC with one intermediate band as a function of the two energy transitions E_1 and E_2 for fully concentrated light $X = 1/Fsun$. The color bar on the right hand side of the contour plot summarizes what efficiencies the various colors represent.

of implementing in any computer language. The general form is the following,

$$g_v(\eta) = \frac{1}{\Gamma(v+1)} \int_0^\infty \frac{E^v dE}{e^{E-\eta} - 1} \tag{2.15}$$

where $\Gamma(v)$ is the Gamma function and v is an integer. Now it may be shown that $g_v(\eta)$ (when $\eta < 0$) can be turned into [20, 21, 22]

$$g_v(\eta) = \sum_{r=1}^{\infty} \frac{e^{r\eta}}{r^{v+1}}. \tag{2.16}$$

This series can be evaluated by using only a finite number of terms and bounding the error [19].

$$g_v(\eta) = \sum_{r=1}^{m-1} \frac{e^{r\eta}}{r^{v+1}} + \Delta \tag{2.17}$$

$$\Delta = \frac{e^{m\eta}}{m^{v+1}[1 - (\frac{m}{m+1})^{v+1} e^{\eta}]} \tag{2.18}$$

In the present analysis, the flux integral has finite limits so this integral expansion is incomplete. Using a similar expansion, the integral

$$I_v(\eta, \epsilon) = \frac{1}{\Gamma(v+1)} \int_\epsilon^\infty \frac{E^v dE}{e^{E-\eta} - 1} \tag{2.19}$$

can be turned into

$$I_v(\eta, \epsilon) = \sum_{r=1}^{\infty} \frac{e^{r(\eta-\epsilon)}}{\Gamma(v+1)} (\frac{\epsilon^v}{r} + \frac{v\epsilon^{v-1}}{r^2} + \frac{v(v-1)\epsilon^{v-2}}{r^3} + \ldots) \tag{2.20}$$

Noticing that there are finite number of terms, it is compactly put into the form

$$I_v(\eta, \epsilon) = \sum_{k=0}^{v} \frac{\epsilon^{v-k} g_k(\eta - \epsilon)}{(v-k)!} \tag{2.21}$$

where $g_k(\eta-\epsilon)$ is equation Eq. 2.16. If both limits are finite, as is the case with some of the integrals evaluated, then this summation can be extended to the following form,

$$I_v(\eta,\epsilon_1,\epsilon_2)=\sum_{l}^{2}\sum_{k=0}^{v}(-1)^{l+1}\frac{\epsilon_l^{v-k}g_k(\eta-\epsilon_l)}{(v-k)!}. \tag{2.22}$$

These approximations to the integrals can be implemented in code with minimal difficulty. As an example, we evaluate the integral $N(E_g,\infty,T_a,qV)$. First, the integral must be transformed into a form similar to Eq. 2.21. This is done with the substitution $x=E/k_BT_a$ and $\eta=qV/k_BT_a$.

$$N(E_g,\infty,T_a,qV)=\frac{2\pi(kT_a)^3}{h^3c^2}\int_{E/kT_a}^{\infty}\frac{x^2dx}{e^{x-\eta}-1}$$

Once in this form, the integral turns into

$$N(E_g,\infty,T_a,qV)=\frac{2\pi(kT_a)^3}{h^3c^2}\cdot\Gamma(3)\cdot I_2(qV/kT_a,E/kT_a) \tag{2.23}$$

and now this is in a form that can be utilized by a computer.

We now will now turn our discussion to the algorithm used to find the chemical potentials. Since the intermediate band of the IBSC is thermally isolated, the the current entering the intermediate band must equal the current leaving the intermediate band. This was the constraint equation introduced in the analysis, that when combined with the Eq. 2.12, all the chemical potentials can be found for every band configuration. On the computer, Eq. 2.21 and 2.22 are substituted into the constraint equation involving the current density. However, this isn't your typical equation that can be solved algebraically since it involves many terms from the summation and exponentials. To remedy, all the terms are put on one side of the equation so that it becomes a root finding exercise using one dimensional numerical techniques. Newton, bisection, or secant methods are all appropriate to implement. We chose to use Newton's method because it can be extended to N-dimensions, which will be useful in solving the constraint equations for the N-intermediate band IBSC.

The idea of the Newton's method in one-dimension is to find the local tangent line for the function $y=f(x)$ and use the zero of this line $(y(x)=0)$ as the next approximation to the zero of the function [23]. Mathematically the tangent line is

$$y(x)=f(x_i)+f_x(x_i)(x-x_i)$$

where $f_x(x_i)$ is the derivative of the function evaluated at the guess x_i. Substituting $y(x)=0$ will give the next iteration.

$$x_{i+1}=x_i-\frac{f(x_i)}{f_x(x_i)} \tag{2.24}$$

If the derivative cannot be analytically determined (as in our case) it is done numerically:

$$f_x(x_i)=\frac{f(x_i+\delta)-f(x_i)}{\delta}. \tag{2.25}$$

Extending Newton's method to N-dimensions is straightforward, x coordinate is replaced by the vector $\mathbf{x}=(x_1,x_2,\ldots,x_n)$, the guess x_i is replaced by the vector $\mathbf{x}_i=(x_{1i},x_{2i},\ldots,x_{ni})$, the function f is repaced by the vector $\mathbf{f}=(f_1,f_2,\ldots,f_n)$, and the derivative f_x is replaced by $\nabla=(\frac{\partial}{\partial x_1},\frac{\partial}{\partial x_2},\ldots,\frac{\partial}{\partial x_n})$. In two-dimensions, using the coordinates x and y for simplicity, we get two simultaneous equations

$$f_1(x_i,y_i)+\frac{\partial f_1(x_i,y_i)}{\partial x}(x-x_i)+\frac{\partial f_1(x_i,y_i)}{\partial y}(y-y_i)=0$$

$$f_2(x_i, y_i) + \frac{\partial f_2(x_i, y_i)}{\partial x}(x - x_i) + \frac{\partial f_2(x_i, y_i)}{\partial y}(y - y_i) = 0 \tag{2.26}$$

and they can be solved using Cramer's rule.

$$\begin{bmatrix} \frac{\partial f_1(x_i,y_i)}{\partial x} & \frac{\partial f_1(x_i,y_i)}{\partial y} \\ \frac{\partial f_2(x_i,y_i)}{\partial x} & \frac{\partial f_2(x_i,y_i)}{\partial y} \end{bmatrix} \cdot \begin{bmatrix} x - x_i \\ y - y_i \end{bmatrix} = \begin{bmatrix} -f_1(x_i, y_i) \\ -f_2(x_i, y_i) \end{bmatrix} \tag{2.27}$$

$$x - x_i = \frac{f_2 \frac{\partial f_1}{\partial y} - f_1 \frac{\partial f_2}{\partial y}}{\frac{\partial f_1}{\partial x} \cdot \frac{\partial f_2}{\partial y} - \frac{\partial f_1}{\partial y} \cdot \frac{\partial f_2}{\partial x}} \tag{2.28}$$

$$y - y_i = \frac{f_1 \frac{\partial f_2}{\partial x} - f_2 \frac{\partial f_1}{\partial x}}{\frac{\partial f_1}{\partial x} \cdot \frac{\partial f_2}{\partial y} - \frac{\partial f_1}{\partial y} \cdot \frac{\partial f_2}{\partial x}} \tag{2.29}$$

$$x_{i+1} = x_i + (x - x_i) \tag{2.30}$$

$$y_{i+1} = y_i + (y - y_i) \tag{2.31}$$

2.2 Two Intermediate Band Detailed Balance Analysis

In an IBSC device where there are two intermediate bands located between the conduction and valence band (see Fig. 2.1), there is a total of six upward energy transitions in this device E_1, E_2, E_3, E_4, E_5 and E_g: E_1 represents valence to first intermediate band, E_2 represents first intermediate to second intermediate band, E_3 represents second intermediate to conduction band, E_4 represents valence to second intermediate band, E_5 represents first intermediate band to conduction band, and E_g represents the conventional band gap between the valence and conduction band. The three intermediate transitions E_1, E_2, and E_3 are independent of each other, while the three other transitions are a function of the intermediate transitions.

$$E_4 = E_1 + E_2 \tag{2.32}$$

$$E_5 = E_2 + E_3 \tag{2.33}$$

$$E_g = E_1 + E_2 + E_3 \tag{2.34}$$

As with the IBSC with one intermediate band, the Fermi energy level must split into the number of total bands or four separate chemical potentials μ_n, μ_{i_1}, μ_{i_2}, and μ_p that represent steady state solutions that allow treatment of the carrier dynamics in the conduction, first intermediate, second intermediate, and valence band respectively.

$$f_v = \frac{1}{e^{(E_v - \mu_p)/k_B T} + 1} \tag{2.35}$$

$$f_c = \frac{1}{e^{(E_c - \mu_n)/k_B T} + 1} \tag{2.36}$$

$$f_{I_1} = \frac{1}{e^{(E_{I_1} - \mu_{i_1})/k_B T} + 1} \tag{2.37}$$

$$f_{I_2} = \frac{1}{e^{(E_{I_2} - \mu_{i_2})/k_B T} + 1} \tag{2.38}$$

Using the same performance characteristics and assumptions of the IBSC containing one intermediate band, we can perform limiting efficiency calculations on the IBSC containing two intermediate bands thereby extending the analysis. The only difference with the analysis is that there are more variables and more equations to solve. For clarity, we list the assumptions for the IBSC again.

1. The solar cell absorbs blackbody radiation at a temperature of T_s =6000°K and ambient T_a =300°K and emits blackbody radiation at ambient T_a =300°K;

2. Only radiative transitions occur between the bands;
3. All photons above the lowest energy gap are absorbed and no high energy photon is used in a low energy process;
4. Carrier mobility is infinite and as a consequence, the quasi-Fermi energy levels are constant throughout the cell;
5. Only one electron-hole pair is created per photon;
6. A perfect mirror is located on the back of the cell so that radiation makes a double pass through the cell and can only escape through the front area of illumination;
7. No carriers are extracted from the intermediate band(s);
8. The net photon flux (number of incident minus number of emitted photons) is equal to the number of charge carrier pairs collected at the contacts (detailed balance assumption).

As before, full absorption is achieved and the emitted photon population from the device in each mode is in thermal equilibrium. If we make the assumption that $E_1 \leq E_2 \leq E_3 \leq E_4 \leq E_5 \leq E_g$, results are independent of this configuration, then

$$n(\epsilon) = \left\{ \begin{array}{ll} 0, & 0 \leq \epsilon < E_1 \\ \frac{1}{e^{(\epsilon - \mu_{I_1 v})/k_B T} - 1}, & E_1 \leq \epsilon < E_2 \\ \frac{1}{e^{(\epsilon - \mu_{I_2 I_1})/k_B T} - 1}, & E_2 \leq \epsilon < E_3 \\ \frac{1}{e^{(\epsilon - \mu_{c I_2})/k_B T} - 1}, & E_3 \leq \epsilon < E_4 \\ \frac{1}{e^{(\epsilon - \mu_{I_2 v})/k_B T} - 1}, & E_4 \leq \epsilon < E_5 \\ \frac{1}{e^{(\epsilon - \mu_{c I_1})/k_B T} - 1}, & E_5 \leq \epsilon < E_g \\ \frac{1}{e^{(\epsilon - \mu_{cv})/k_B T} - 1}, & \epsilon \geq E_g \end{array} \right\} \tag{2.39}$$

where the chemical potentials are the respective differences between the quasi-Fermi levels. For simplicity we label the valence band v, first intermediate band I_1, second intermediate band I_2, and the conduction band c. The current density will be equal to the charge carriers excited to the conduction band

$$\begin{aligned} &J(E_1, E_2, E_3, T_s, T_a, X, V) = \\ &q[XF_{sun}N(E_g, \infty, T_s, 0) + (1 - XF_{sun})N(E_g, \infty, T_a, 0) - F(\pi/2)N(E_g, \infty, T_a, qV)] \\ &+q[XF_{sun}N(E_5, E_g, T_s, 0) + (1 - XF_{sun})N(E_5, E_g, T_a, 0) - F(\pi/2)N(E_5, E_q, T_a, \mu_{cI_1})] \\ &+q[XF_{sun}N(E_3, E_4, T_s, 0) + (1 - XF_{sun})N(E_3, E_4, T_a, 0) - F(\pi/2)N(E_3, E_4, T_a, \mu_{cI_2})] \end{aligned} \tag{2.40}$$

where we have made use of Eq. 1.12, the geometric factor $F(\theta)$, and the concentration factor X. The terms in the first bracket on the right hand side of the equation represent the current density generated from the promotion of electrons from the valence band to the conduction band less recombination events from the reverse transition, the terms in the second bracket represent the current density generated from the promotion of electrons from the first intermediate band to the conduction band less recombination events from the reverse transition, and the terms in the last bracket represent the current density generated from the promotion of electrons from the second intermediate band to the conduction band less recombination events for the reverse transition. In each of the bracketed terms, the IBSC absorbs radiation from the sun at the characteristic temperature T_s over the angular range $0 < \theta < 0.26^\circ$ and from ambient at the characteristic temperature T_a over the angular range $0.26^\circ < \theta < \pi/2$, while the IBSC emits radiation at the characteristic temperature T_a and characteristic uniform chemical potential over the angular range $0 < \theta < \pi/2$.

For each energy configuration (E_1, E_2, E_3), there exists a voltage V_m that maximizes the IBSC power density output $V_m \cdot J_m(E_1, E_2, E_3, T_s, T_a, X, V)$. However, as with the one intermediate

band IBSC, the current density contains chemical potentials for which we need to solve for before calculating. To find all the chemical potentials for a particular energy configuration, we employ the constraints:

1. The current entering intermediate band I_1 must equal the current leaving the intermediate band I_1 ($J_{vI_1} = J_{I_1I_2} + J_{I_2c}$) and the current entering intermediate band I_2 must equal the current leaving the intermediate band I_2 ($J_{I_1I_2} + J_{vI_2} = J_{I_2c}$).

2. The chemical potential μ_{I_2v} must equal the sum of the chemical potentials $\mu_{I_1v} + \mu_{I_2I_1}$. The chemical potential μ_{cI_1} must equal the sum of the chemical potentials $\mu_{I_2I_1} + \mu_{cI_2}$. Finally and as before, the chemical potential $\mu_{cv} = qV$ must equal the sum of the quasi-Fermi levels $\mu_{I_1v} + \mu_{I_2I_1} + \mu_{cI_2}$.

where we have simplified the current density notation such that J_{vI_1} is equal to

$$q[XF_{sun}N(E_1, E_2, T_s, 0) + (1 - XF_{sun})N(E_1, E_2, T_a, 0) - F(\pi/2)N(E_1, E_2, T_a, \mu_{vI_1})]$$

and so on. The same general approach to find all the chemical potentials (total of 6) is used except this time we employ Newton's method in two dimensions and numerically solve the simultaneous Eqs. 2.26 using Cramer's rule.

Following the prescription as outlined in the previous sections, we have found the limiting efficiency of the IBSC by varying the energy transitions E_1 and E_2 above the valence band for unconcentrated $X = 1$ and fully concentrated $X = 1/F_{sun}$ light. Fully concentrated light has a maximum efficiency of 72.4% for the energy transitions of 0.59 eV, 0.93 eV, and 1.05 eV for a band gap $E_g = 2.57$ eV, while unconcentrated light has a maximum efficiency of 52.1% for the energy transitions of 0.85 eV, 1.20 eV, and 1.43 eV for a band gap $E_g = 3.48$ eV. As the concentration factor increases, so does the limiting efficiency up to the maximum of 72.4%, a result that is directionally similar to both the conventional solar conversion device and IBSC with one intermediate band. We should note that the two patterns that we have previously identified by considering the limiting efficiency calculations is true for the two intermediate band IBSC: (1) efficiency is proportional to light concentration and (2) the large band gap E_g is inversely proportional to light concentration.

As we compare the performance of the two intermediate band IBSC model to the conventional solar conversion model, it is evident that conversion efficiency has again significantly improved. For unconcentrated light, performance has increased from 31% to 52.1% which is a 40.5% increase in efficiency. For fully concentrated light, performance has increased from 41% to 72.4% which is a 43.4% increase in efficiency. When we compare the performance of the two intermediate band IBSC to the one intermediate band IBSC, conversion efficiency improves. For unconcentrated light, performance has increased from 46.8% to 52.1% which is a 10.2% increase in efficiency. For fully concentrated light, performance has increased from 63.2% to 72.4% which is a 12.7% increase in efficiency.

As with the one intermediate band IBSC, it is important to investigate the sensitivity of efficiency as a function of the two independent energy transitions E_1 and E_2 with the band gap E_g of the device held fixed for unconcentrated and fully concentrated light. We do this for the band gap $E_g = 3.48$ eV and $E_g = 2.57$ eV that corresponds to the maximum efficiency of both cases (see Figs. 2.5 and 2.6). For unconcentrated light, the maximum efficiency of 52.1% occurs at the energy band transitions $E_1 = 0.85$ eV and $E_2 = 1.20$ eV for a band gap of $E_g = 3.48$ eV but Fig. 2.5 shows that an efficiency of $\geq 50\%$ can occur at various band configurations. From Fig. 2.5, the IBSC can achieve a limiting efficiency $\geq 50\%$ for band configurations ranging between 0.79 eV $\leq E_1 \leq$ 0.88 eV and 1.16 eV $\leq E_2 \leq$ 1.25 eV. It is possible to fit a polynomial to find the interval of $E_2 = [a(E_1), b(E_1)]$ that results in an efficiency $\geq 50\%$ as we have done for the one intermediate band IBSC. However, due to an additional degree of freedom, each band gap will have its own efficiency profile in terms of the two energies E_1 and E_2. The analytic expressions describing a desired interval for one band gap will not be the same for another band gap, making this type of calculation less useful.

For fully concentrated light, the maximum efficiency of 72.4% occurs at the energy band transitions $E_1 = 0.59$ eV and $E_2 = 0.93$ eV for a band gap of $E_g = 2.57$ eV but Fig. 2.6 shows that an efficiency of $\geq 70\%$ can occur at various band configurations. From Fig. 2.6, the IBSC can achieve

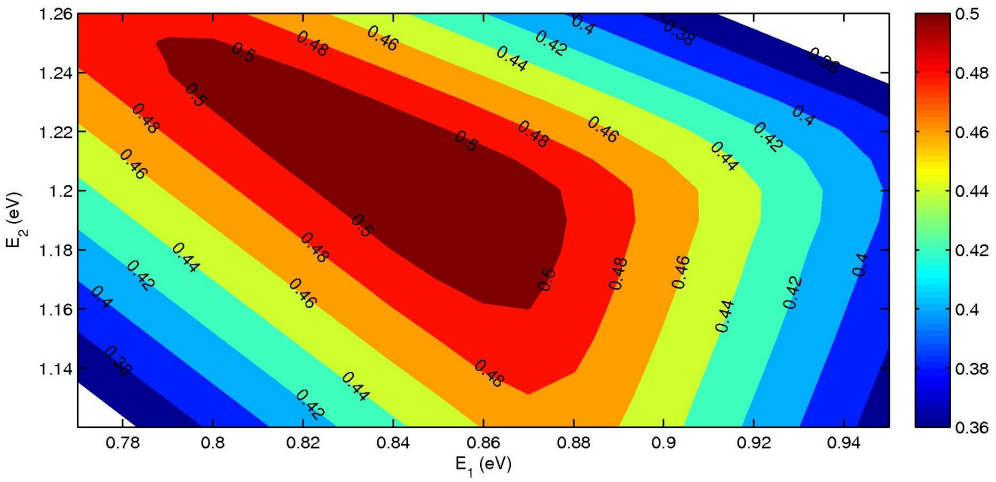


Figure 2.5: Limiting efficiency of a two intermediate band IBSC with band gap $E_g = 3.48$ eV as a function of the two energy transitions E_1 and E_2 for unconcentrated light $X = 1$. The color bar on the right hand side of the contour plot summarizes what efficiencies the various colors represent.

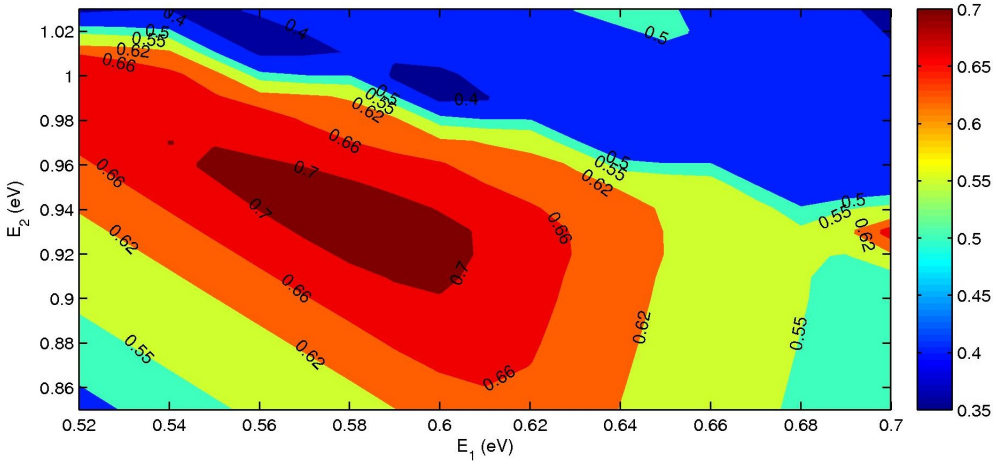


Figure 2.6: Limiting efficiency of a two intermediate band IBSC with band gap $E_g = 2.57$ eV as a function of the two energy transitions E_1 and E_2 for fully concentrated light $X = 1/Fsun$. The color bar on the right hand side of the contour plot summarizes what efficiencies the various colors represent.

a limiting efficiency $\geq 70\%$ for band configurations ranging between 0.54 eV $\leq E_1 \leq$ 0.61 eV and 0.91 eV $\leq E_2 \leq$ 0.97 eV.

2.3 Remarks

In this chapter, we have introduced a theoretical device called the intermediate band solar cell that would contain intermediate bands between the valence and conduction bands aimed at increasing conversion efficiency. The intermediate bands allow for low energy photons to promote charge carriers in a stepwise manner to the conduction band that would normally be lost as heat or not absorbed by the conversion device. In addition, incoming photons would be better matched with energy transitions between bands thereby reducing charge carrier thermalization. We described that proper operation of the IBSC would require that no charge carriers are allowed to be extracted from the intermediate bands. This is important for two reasons. First, this would deplete the carrier density available for excitation to higher bands, including the conduction band. Second, charge carriers are extracted from intermediate bands at lower potential differences than from the conduction band. This would reduce the power output of the IBSC and thus the conversion efficiency. Since carriers would only be extracted from the valence and conduction band, the IBSC would operate similar to the conventional solar cell in the sense that the operating voltage would be proportional to the band gap.

We then performed limiting efficiency calculations on the IBSC and compared them to the detailed balance of the conventional solar cell. For an IBSC that contains one intermediate band, we calculated a limiting efficiency of 63.2% when the largest band gap is $E_g = 1.93$ eV, while unconcentrated light has a maximum efficiency of 46.8% when the largest band gap is $E_g = 2.40$ eV. For an IBSC that contains two intermediate bands, we calculated a limiting efficiency of 72.4% when the largest band gap is $E_g = 2.57$ eV, while unconcentrated light has a maximum efficiency of 52.1% when the largest band gap is $E_g = 3.48$ eV. These efficiencies are significant because of the potential increase in efficiency of an IBSC over the conventional solar conversion device. By rethinking the design and operation, significant conversion improvements were realized and this is the motivating factor for further research in this type of structure. This is the first step to design a real world solar conversion device having an optimal intermediate band structure.

Chapter 3: Quantum Dot Intermediate Band Solar Cell

In the previous chapters, we have have studied the limiting efficiency of various solar conversion devices and showed how efficiency responds to the inclusion of intermediate bands placed in the forbidden band gap of a normal semiconductor. We described how the presumed operation of the IBSC leads to the efficiencies that are far superior to the conventional conversion device. The increases can be attributed to utilization of lower energy photons and energy transitions that better match the solar spectrum. For an IBSC that contains one intermediate band, we calculated a limiting efficiency of 63.2% under fully concentrated light and 46.8% under unconcentrated light. For an IBSC that contains two intermediate bands, we calculated a limiting efficiency of 72.4% under fully concentrated light and 52.1% under unconcentrated light. In addition, we found the energy band configurations that correspond to the limiting efficiency and performed sensitivity analyses that showed slight variations do not affect performance. However, we made no mention of how a real device could be designed and operated such that its behavior would exhibit that of an IBSC.

With the advance in microfabrication technology, it is possible to coherently grow two or more dissimilar semiconductors that force quantum confinement for charge carriers. These types of structures are called heterostructers [10]. Heterostructers that confine charge carriers in one direction, two directions, and three directions are called quantum wells, quantum wires, and quantum dots respectively. In this chapter, we introduce a device based on quantum dot technology as a physical realization of the IBSC called the quantum dot intermediate band solar cell (QD-IBSC). We begin in Sec. 3.1 by describing the types of heterostructures and how confinement leads to discrete energy levels that can appear in the forbidden band gap of a semiconductor. We expand this idea in Sec. 3.2 by introducing the QD-IBSC and explain how it can lead to an IBSC device. The QD-IBSC is further examined in Sec. 3.3 by outlining design criteria necessary to achieve the IBSC operation. We show, based on a simplistic model, how quantum dot materials and parameters can be identified that closely match the intermediate band energies that provide the maximum theoretical efficiency. We conclude with Sec. 3.4.

3.1 Heterostructures

A heterostructure consists of two or more different semiconductors that are grown coherently with one common crystal structure [10]. Of real interest in these types of structures are the effects caused by the different band gaps and relative energy alignment with respect to each other. As an example, we consider a heterostructure consisting of compound semiconductors aluminum arsenide and gallium arsenide (AlAs/GaAs) material that alternates along the z-direction (see Fig. 3.1). The band gap of the pure compound semiconductor AlAs is larger than the band gap of the pure compound semiconductor GaAs. In addition to the size of the band gaps, the relative band alignment between the two permits the GaAs conduction band to sit below the conduction band of AlAs and the GaAs valence band to sit above the valence band of AlAs. As a result, there is an energy separation between the GaAs and AlAs valence bands. This is called the valence band offset (see Fig. 3.1). The conduction band offset is defined similarly. The alternating materials and offsets create one dimensional potential wells. Within each well, additional GaAs energy levels below its valence band could lie above the AlAs valence band (energy level v1 in Fig. 3.1). Similarly, energy levels arise between the conduction bands of GaAs and AlAs (energy level c1 and c2 in Fig. 3.1). As the number of quantum wells increases, these energy levels will split and spread into bands. Such bands occur as intermediate bands between the band gap of AlAs in the AlAl/GaAs heterostructure.

In a photovoltaic device, carrier dynamics take place at energies near the conduction band minimum and valence band maximum, see discussion on thermalization in Sec. 1.1. It is these stationary

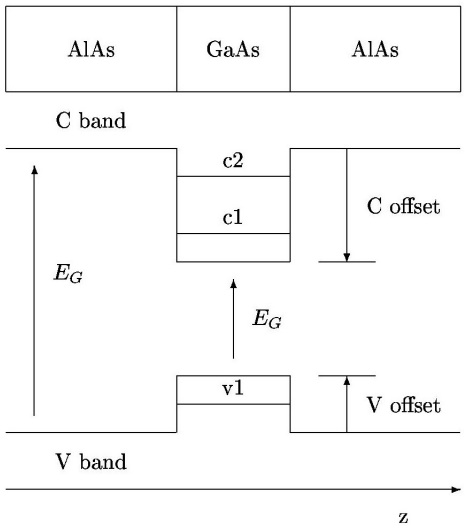


Figure 3.1: The alternating semiconducting material AlAs and GaAs creates a heterostructure. C and V refer to conduction and valence bands, respectively. Since the bandgaps are of different sizes, this mimics the one dimensional potential well. The intermediate energy levels c1, c2, and v1 are in the quantum wells, the offset is the potential well depth.

points that determine the band gap of the semiconductor[1]. If we expand the energy around the extrema, in both bands, as a function of the wavevector $\vec{k}$, a useful approximation emerges that will be used throughout this text in calculations. In the conduction band, the minimum energy is located at $E_c(\vec{k_{0c}})$ and we expand around the point $\vec{k_{0c}}$ such that

$$E_c(\vec{k}) = E_c(\vec{k_{0c}}) + \underbrace{\frac{\partial E_c(\vec{k})}{\partial k}\,|_{k_{0c}}\,(\vec{k}-\vec{k_{0c}})}_{=0} + \frac{1}{2}\frac{\partial^2 E_c(\vec{k})}{\partial k^2}\,|_{k_{0c}}\,(\vec{k}-\vec{k_{0c}})^2 \tag{3.1}$$

$$= E_c(\vec{k_{0c}}) + \frac{\hbar^2(\vec{k}-\vec{k_{0c}})^2}{2m_c^*} \tag{3.2}$$

where we have made use of a parameter m_c^* that has the dimensions of mass. It is defined using the following relation

$$\frac{1}{m_c^*} = \frac{1}{\hbar^2}\frac{\partial^2 E_c(\vec{k})}{\partial k^2}\,|_{k_{0c}} \tag{3.3}$$

and the *effective mass* m_c^* is analogous to the mass of the free electron m but differs through the different forces experienced in the lattice. The effective mass can be greater or less than m depending on the atomic potentials. Near the band edge, m_c^* is often isotropic and can be approximated by a constant [9]. Literature containing a comprehensive review of semiconductor band parameters will list m_c^* in terms of the electron mass, an example is a review by Vurgaftman *et al.* [1] that we have made use of extensively. We make the observation from Eq. 3.2 that the energy of an electron near the conduction band minimum in a lattice is similar to that of a free electron but differs by m_c^*.

[1]There are two types of band gaps that can occur in semiconductors, direct and indirect. A direct band gap is when the momentum $\hbar k$ is the same for both the conduction band minimum and valence band maximum, while an indirect band gap is when the momentum for the conduction band minimum is different than the momentum for the valence band maximum. An electron transition from the valence to conduction band (or reverse transition) for a direct band gap semiconductor only requires a photon. However, an indirect band gap semiconductor band transition requires a photon *and* shift in momentum, supplied by phonons.

In the valence band, we look for an expression for the energy of a hole near the valence band maximum energy $E_v(\vec{k_{0v}})$. Similar to the conduction band, the energy is expanded in terms of the wavevector $\vec{k}$ around the point $\vec{k_{0v}}$ such that

$$E_v(\vec{k}) = E_v(\vec{k_{0v}}) + \underbrace{\frac{\partial E_v(\vec{k})}{\partial k}\,|_{k_{0v}}\,(\vec{k}-\vec{k_{0v}})}_{=0} + \frac{1}{2}\frac{\partial^2 E_v(\vec{k})}{\partial k^2}\,|_{k_{0v}}\,(\vec{k}-\vec{k_{0v}})^2 \tag{3.4}$$

$$= E_v(\vec{k_{0v}}) - \frac{\hbar^2(\vec{k}-\vec{k_{0v}})^2}{2m_v^*} \tag{3.5}$$

where we have made use of the effective mass of a hole in the valence band m_v^*. It is defined using the following relation

$$\frac{1}{m_v^*} = -\frac{1}{\hbar^2}\frac{\partial^2 E_v(\vec{k})}{\partial k^2}\,|_{k_{0v}} \tag{3.6}$$

so that the effective mass is normally positive. Similar to Eq. 3.2, we make the observation that Eq. 3.5 near the valence band maximum in a lattice is similar to that of a free electron but differs by m_v^*. The above equations represent what is called the parabolic band approximation [24].

We should note that, in general, the effective mass in each band is represented by a tensor and anisotropic. However near band extrema, such as $\vec{k_{0c}}$ and $\vec{k_{0v}}$, the effective mass is often isotropic and can be described by one constant. In cases where the conduction band minimum and valence band maximum occur at different $\vec{k}$, i.e. indirect band gap, the effective mass near the extrema is anisotropic and can characterized by two constants: longitudinal and traverse effective mass. For the reasons described below in Sec. 3.3, we will only be considering materials with direct band gaps such that the effective mass is isotropic as represented by one constant. As such, in the equations and derivations that follow, only one constant representing the effective mass is necessary.

Returning to the quantum well heterostructure described above, discrete energy levels occur in the $z-$direction, while there is no confinement in the quantum well in-plane direction. This confines charge carriers in one direction and produces the energy spectrum of the form $E_d+\hbar^2(k_x^2+k_y^2)/2m^*$, where E_d are discrete energies associated with confinement in the $z-$direction, $\hbar k_x$ and $\hbar k_y$ are the in-plane momentum components, and m^* is the charge carrier effective mass. In order to find the discrete energy levels within each band, we substitute the effective mass for the mass in Schrödinger's time-independent equation and solve for the $z-$direction subject to boundary conditions. In Cartesian coordinates, using the effective mass approximation, Schrödinger's equation is

$$\frac{-\hbar^2}{2m^*}\left(\frac{d^2}{dx^2}+\frac{d}{dy^2}+\frac{d^2}{dz^2}+V(z)\right)\Psi(x,y,z) = E\Psi(x,y,z) \tag{3.7}$$

where $V(z)$ is the conduction or valence band offset in the $z-$direction. In order to solve, appropriate continuity conditions are applied at the heterostructure boundary. For a spatially varying mass, as with the effective mass, the wavefunction and its inverse mass derivative should be continuous (see Sec. 4.1 for derivation, specifically Eq. 4.4).

A semiconductor heterostructure that confines charge carriers in two dimensions such that the energy spectrum is of the form $E_{d_x}+E_{d_y}+\hbar^2(k_z^2)/2m^*$, where $E_{d_x}+E_{d_y}$ are discrete energies associated with confinement in the traverse direction and $\hbar^2(k_z^2)/2m^*$ is the longitudinal momentum component, is called a quantum wire. The discrete energy levels are found by solving Eq. 3.7 with a potential $V(x,y)$ that represents the conduction or valence band offset (see Eq. 5.15).

Semiconductor heterostructures that restrict the motion of charge carriers in all three spatial directions are referred to as quantum dots (QDs) [25]. Unlike the quantum well, the QD material is completely surrounded by a material with a larger band gap (barrier material) so the energy spectrum is discrete. If the number of QDs is increased and arranged in a periodic lattice, the energy levels will split and spread out into bands. The width of the band(s) depends on the spacing of the QDs within the lattice and wavefunction overlap. We will revisit this idea over the next few

sections.

Due to the energy levels that appear between the conduction and valence band of the barrier material, heterostructures are considered for the IBSC. All three types of heterostructures introduced above contain the necessary intermediate levels required for the IBSC. However, only one of the three types allows for the proper operation of the IBSC. The quantum well's intermediate levels provide a finite density of states lying between the conduction and intermediate bands because of the term in the energy spectrum $\hbar^2(k_x^2 + k_y^2)/2m^*$. Photogenerated charge carriers will lose energy through thermalization due to the high density of states available in the in-plane direction. The Fermi level will not split into the number of total bands. Rather, the Fermi level will split into two levels similar to the solar conversion device with a single energy gap. The lowest lying intermediate band will act as the conduction band and heterostructure will behave as a single energy gap conversion device. These ideas have been reviewed by Anderson and Luque *et al.* in [26, 27]. Similar to the quantum well, the quantum wire's intermediate levels provide a finite density of states lying between the conduction and intermediate bands because of the term $\hbar^2(k_z^2)/2m^*$ and photogenerated charge carriers will lose energy through thermalization due to the density of states available in the longitudinal direction. Again, the Fermi level will not split into the number of total bands but rather split into two levels.

As opposed to the quantum well and wire, a QD heterostructure will provide zero density of states between the excited bands due to the discrete energy spectrum. With each band thermally isolated, the Fermi level will split into the number of bands and charge carrier concentration in each of the bands will be described by its own chemical potential as required for proper IBSC operation. This will reduce thermalization and is the main reason why QDs are considered for the physical realization of the IBSC.

3.2 Quantum Dot Intermediate Band Solar Cell

The idea of the quantum dot intermediate band solar cell (QD-IBSC) was first proposed by Marti *et al.* [28], where they describe the structure of the design and offer up a proposed QD/barrier system that, when simplistically modeled, would produce one intermediate band in the conduction band offset at the ideal energetic location determined by the detailed balance method outlined in Chap. 2. The QD-IBSC is shown schematically in Fig. 3.2, with the quantum dot material located in-between the p-n junction. The p-n emitters prevent the intermediate band(s) from touching the deposited contacts on the external faces so no current is directly extracted from the intermediate bands in normal operation [29]. As described in Chap. 2, this is a necessary requirement for the proper operation of the IBSC.

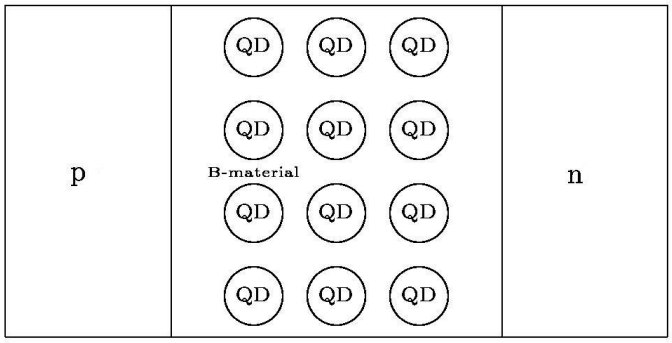


Figure 3.2: An orthogonal projection of the proposed QD-IBSC with the barrier material surrounding the QDs. The QD material is arranged periodically within the barrier material and sandwiched in between the normal p-n junction.

The proposed QD-IBSC by Marti *et al.* is a variation on the p-i-n junction theme used in certain types of solar cells today. The layer of semiconductor material between p and n, instead of being

left undoped or *intrinsic*, contains a periodic array of QDs embedded within the intrinsic, or *barrier* material. It is important for the size and spacing of the quantum dots to be uniform. This helps establish well-placed intermediate band boundaries. If the dots are organized in a random array or are diverse in size, the energy levels will tend to be irregularly spaced throughout the offsets due to symmetry breaking [30]. The intermediate bands would not be created, rather single energy levels varied throughout the band gap aiding in recombination via thermalization.

One way to grow uniform coherent QD arrays is through the Stanski and Krastanov (S&K) method, which leads to self-organized growth [31, 32, 33, 34]. A requirement to achieve this growth is a lattice constant mismatch, Δ_{lc}, between the quantum dot and barrier material. The formulation of the quantum dots are a result of the Δ_{lc}: as the quantum dot material is deposited on the surface of barrier material, the quantum dot material will compress to fit the smaller barrier lattice and 2D growth will continue until it becomes energetically favorable for quantum dots (islands) to spontaneously form. The layer thickness at which 2D-3D growth occurs is called the critical thickness and depends on the Δ_{lc} [35]. The QD characteristics such as size and shape are highly dependent on certain parameters such as growth temperature and growth interruptions. Outside the scope of this thesis is a discussion of the methods that are used to grow the self-assembled QDs. There are various growth techniques, such as Metal Organic Chemical Vapor Deposition (MOCVP) and Molecular Beam Epitaxy (MBE), with each one very different from the next and constantly evolving. We note that QD growth technology is available and can produce highly uniform QD arrays. One main requirement for doing so is a lattice mismatch between the quantum dot and barrier material.

Marti *et al.* made the assumption that the geometry of the QD is spherical and is characterized by a QD radius, a, measured in Angstroms, Å. In addition, for simplicity, they assumed that the energy corresponding to the top of the valence band is the same both in the barrier and the QD material. This means there is no valence band offset and the only confining potential occurs at the conduction band offset. Under these assumptions, they calculated the energy levels in the QD using the effective mass approximation. The charge carrier is confined to the spherical QD of radius a such that the potential is

$$V(r) = \left\{ \begin{array}{cc} -V_0, & 0 \leq r < a \\ 0, & r > a \end{array} \right\} \tag{3.8}$$

where $-V_0$ is the conduction band offset. We also assume that the charge carrier's effective mass varies between the interior and exterior of the well as follows [9]:

$$m(r)^* = \left\{ \begin{array}{cc} m_D, & 0 \leq r < a \\ m_B, & r > a \end{array} \right\} \tag{3.9}$$

The energy levels in a spherical well are found by factoring the wavefunction into a radial and angular part ($R_{nl}(r)Y_m^l(\theta,\phi)$) and solving the radial equation subject to the boundary conditions [15]:

$$R(r) \text{ is finite as } \quad \text{as} \quad r \to 0 \quad \text{and} \quad R(r) \to 0 \quad \text{as} \quad r \to \infty \tag{3.10}$$

In the interior of the well the solutions are spherical Bessel functions $R(r) \simeq j_l(kr)$ of the first kind, with $E + V_0 = \hbar^2 k^2/2m_D > 0$. The second solutions ($y_l$) are rejected because they diverge at the origin. For $r > a$ and $E < 0$ the solutions are spherical Bessel functions of imaginary argument $R(r) \simeq j_l(i\kappa r)$, with $E = \hbar^2(i\kappa)^2/2m_B < 0$. We use the solution that decreases exponentially to zero as $r \to \infty$. The solutions are determined by enforcing continuity at $r = a$ [36]:

$$R_{\text{inside}}(r)|_{r=a} = R_{\text{outside}}(r)|_{r=a} \quad \text{and} \quad \frac{1}{m_D}\frac{R_{\text{inside}}(r)}{dr}|_{r=a} = \frac{1}{m_B}\frac{R_{\text{outside}}(r)}{dr}|_{r=a} \tag{3.11}$$

The bound state energy eigenvalues are labeled $E_{n,l}$, where n is the number of radial nodes and l is the orbital angular momentum. The ground state energy is $E_{0,0}$.

Using the outlined approach, Marti *et al.* found that the ternary alloy $Al_{0.40}Ga_{0.60}As$ could be used as the barrier material and the ternary alloy $In_{0.42}Ga_{0.58}As$ as the QD material. Band parameters for the barrier material are as follows: a band gap of 1.95 eV, an effective mass of

$0.096m_0$ expressed as the resting mass of an electron m_0, and a lattice constant of 5.5654 Å. Band parameters for the QD material are as follows: a band gap of 0.87 eV, an effective mass of $0.045m_0$ expressed as the rest mass of an electron m_0, and a lattice constant of 5.7060 Å. The ground state energy level $E_{0,0}$ for a QD radius of $a = 39$ Ålies 1.24 eV above the valence band, which is the optimal position for achieving 63.2% conversion efficiency as determined from the detailed balance argument. In addition, there is the necessary lattice mismatch between the QD and barrier material needed to manufacture the QD array.

Although the proposed QD-IBSC materials seemingly have the desired parameters identified for an IBSC under fully concentrated light, there exist unjustified assumptions that should be noted. The valence band offset in the present QD heterostructure is not negligible, rather the depth is 0.37 eV. This confining potential will support energy levels, in fact it will support many energy levels because holes are generally heavy. This could potentially be a problem because as these energy levels spread out into bands, some will overlap or those next to the barriers valence band will merge with that band. This result might reduce the efficiency through unwanted transitions or shrink the barrier energy gap due to bands merging with the barriers valence band. A second assumption is that the conduction band offset will only support one energy level. However, the conduction band offset will support not only the ground state energy level but also support two additional energy levels $E_{1,1}$ and $E_{2,2}$. Additional energy levels will affect efficiency. Similar to the valence band multiple energy level discussion, when these energy levels spread out into bands they might overlap with each other or merge into the barrier's conduction band. Efficiency could be reduced. If, for example, one of the bands overlaps with the conduction band, the energy gap of the barrier material will decrease. This will degrade the cell's operating voltage and decrease the power output of the cell. If the additional bands are optimally placed as we saw in Chap. 2, they could be used to enhance efficiency. As such, it is important to calculate the placement and width of the bands. The tight binding method described by Slater *et al.* is a suitable method that is used to calculate the width of the bands throughout the rest of this thesis [37].

3.3 Design Considerations and Results

Levy *et al.* also found potential QD-IBSC materials that contain an optimally placed intermediate band with a detailed balance conversion efficiency of 60% [38]. As part of their design criteria, they selected QD-IBSC materials that have a negligible valence band offset. However, using the model described in Sec. 3.2 for locating the energy levels and realistic size of quantum dots, two intermediate bands will most likely emerge. This second band, if strategically placed, will increase the maximum efficiency of the solar cell from 63.2% [16] (single intermediate band) to 72.4% (see Fig. 2.6) for an IBSC under full concentration. When searching for materials, this additional band should be taken advantage of to maximize the QD-IBSC capability. Using the procedure outlined in Sec. 3.2, the bound state energy levels $E_{n,l}$ obtained for typical quantum dot parameters are organized as follows: $E_{0,0} < E_{0,1} < E_{0,2} < E_{1,0} \cdots$. This is an important result for two reasons:

- The first intermediate energy level is found when $n = 0, l = 0$ and the second intermediate energy level is found when $n = 0, l = 1$.
- The quantum dot will support two intermediate energy levels when the second excited state, $n = 0, l = 2$, is not bound but first excited state, $n = 0, l = 1$, is bound.

This knowledge is useful when searching through numerous materials; only the values $n = 0$ and $l = 0, 1,$ and 2 are used in calculations to find energy levels.

For an efficiency of an IBSC under full concentration of at least 70%, the barrier energy gap must be greater than 2.0 eV. This is the first design criterion and begins to narrow down possible barrier materials.

As mentioned, a valence band offset might form and the confining potential will support many energy levels. This result might reduce the efficiency through unwanted transitions or shrink the barrier energy gap due to bands merging with the barrier's valence band. Therefore, the materials determined have a negligible valence band offset in order to eliminate these possibilities. This will

assure that no minibands are created outside those that are known and calculated in the conduction band offset.

As a final design criterion, it is desirable to produce solar cells as thin as possible. Thinner solar cells require less material and will reduce manufacturing costs. One way to reduce the thickness of the cell is to use direct band gap materials so the absorption coefficient is strong. Only direct band gap materials are used in this search.

We highlight potential QD-IBSC materials found that satisfy the design considerations and optimized two intermediate energy levels to achieve a theoretical efficiency greater than 70%. The two energy levels located in the conduction band offset are found using a computer program utilizing the method described by Sec. 3.2 in conjunction with band parameters (based on the review from Vurgaftman *et al.* [1]), all of which were calculated at room temperature, 300°K. The barrier and quantum dot materials, along with the energy transitions, lattice mismatch, quantum dot radius, and efficiency are given in Tab. 3.1. This table lists possible QD-IBSC with efficiencies in excess of

Table 3.1: Barrier and quantum dot materials (QD) that produce an efficiency, η, greater than 70% and have two intermediate bands. The energy transitions E_1, E_2, and E_3 refer to those in Fig. 2.1.

Barrier material QD material	$AlAs_{0.17}Sb_{0.83}$ $InP_{0.35}Sb_{0.65}$	$AlP_{0.05}Sb_{0.95}$ $GaP_{0.37}Sb_{0.63}$	$AlAs_{0.05}Sb_{0.95}$ $GaAs_{0.72}Sb_{0.28}$
QD radius (Å)	35	34	33
E_1 (eV)	0.73	1.23	1.31
E_2 (eV)	1.31	0.43	0.43
E_3 (eV)	0.26	0.57	0.55
η(%)	70.0	72.0	70.0
Δ_{LC}(%)	3.45	4.30	5.78

70%.

However, Tab. 3.1 does not give any information about the dependence of efficiency on the variation of the energy levels. The sensitivity of efficiency as a function of the energy transitions was investigated. The band gap of the barrier is held fixed and efficiency is calculated at various energy transitions E_1 and E_2. For illustrative purposes, the barrier/QD material AlPSb/GaPSb was used and a contour plot was generated (see Fig. 3.3). Efficiencies greater than 70% can be attained as long as the energy transitions are within, roughly speaking, an ellipse that ranges between 1.21 eV $< E_1 <$ 1.29 eV and 0.37 eV $< E_2 <$ 0.45 eV. Outside of these ranges, the efficiency drops below the desired level and in some cases below the one intermediate band efficiency of 63.2%. As a consequence, there is a design trade off. Instead of looking to produce the maximum efficiency of a given system, a slightly less efficient system could be designed that is more robust under parameter perturbation. As an example, consider AlPSb/GaPSb which might be designed to $E_1 = 1.25$ eV and $E_2 = 0.41$ eV. This allows for the most deviation, E_1 and E_2 can vary by ± 0.04 eV, while keeping the efficiency greater than 70%.

The transition energies will spread out into bands as the quantum dots are grown. As mentioned, it is important that the intermediate bands do not overlap with each other or with the conduction/valence bands. Qualitatively, the closer the quantum dots are spaced, the larger the width of the intermediate energy bands due to the overlap of their wavefunctions. Quantitatively, there should be some minimum distance between the quantum dots in order to prevent the bands from overlapping.

Using tight-binding theory and assuming the quantum dots are arranged in a simple cubic lattice, minimum distances are determined. This distance is measured from the center of one quantum dot to the center of the next quantum dot. Table 3.2 displays the minimum distance, D, that is required to prevent an overlap of the minibands and the widths of minibands appearing in the QD, Δs and Δp, at the distance D. For the barrier/QD materials AlPSb/GaPSb and AlAsSb/GaAsSb, the

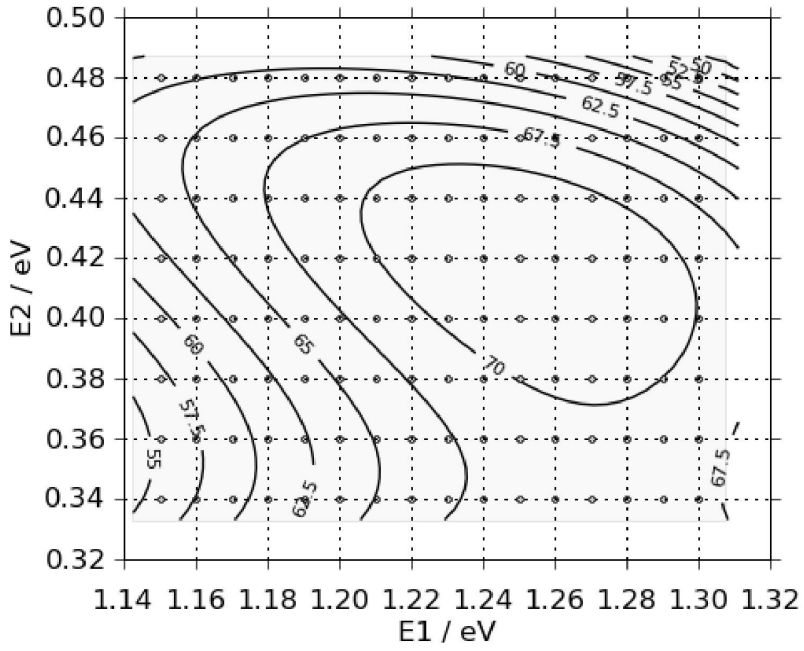


Figure 3.3: Contour plot displaying calculated efficiency of the 4 band solar cell with an energy gap of $E_g = 2.23$ eV. The two energy transitions, E1 and E2, are those referred to in Fig. 2.1.

minibands will not overlap even when the QDs touch each other in the lattice. In this case, the minimum distance is determined by the physical constraint of the QDs rather than the prevention of miniband overlap.

Table 3.2: Maximum bandwidth associated with the two intermediate energy levels in a simple cubic lattice. Δs is this bandwidth associated with the first energy level, Δp is the bandwidth associated with the second energy level, and D is the minimum distance as measured from the center of one quantum dot to center of next quantum dot needed to prevent overlapping.

Barrier QD	$AlAs_{0.17}Sb_{0.83}$ $InP_{0.35}Sb_{0.65}$	$AlP_{0.05}Sb_{0.95}$ $GaP_{0.37}Sb_{0.63}$	$AlAs_{0.05}Sb_{0.95}$ $GaAs_{0.72}Sb_{0.28}$
QD radius (Å)	35	34	33
D (Å)	77	68	66
Δs (eV)	0.033	0.129	0.133
Δp (eV)	0.38	0.287	0.296

Although materials were found that produce the intermediate bands necessary to achieve an efficiency of greater than 70%, these results are based on certain restrictive assumptions: the shape of the quantum dot is spherical; the quantum dot lattice is cubic; the Hamiltonian includes only the offset potential; and the QD-IBSC is under full concentration. It remains to be determined how the efficiency degrades as these assumptions are analyzed and possibly relaxed.

Furthermore, intermediate bands should be approximately half-filled with electrons in order to receive electrons from lower energy bands and pump electrons to higher energy bands [39]. This condition might not possible if there is more than one intermediate band [40]. For those intermediate bands that are mostly empty, light absorption would be weak. A detailed study on how this affects the 4-band IBSC efficiency and possible solutions should be undertaken.

3.4 Remarks

In this chapter, we have studied a possible path to the IBSC through the use of QD technology. Other types of quantum heterostructures, like the quantum well or wire, were disregarded because of the finite number of states available in between minibands and do not satisfy the IBSC requirement that each band's charge carrier concentration must be described by its own quasi-Fermi level under normal operation. We described the QD-IBSC structure as a variation on the p-i-n solar cell theme by asserting that QDs embedded into barrier material would replace the intrinsic material so current would not be extracted from the formed intermediate band(s). The QDs must be arranged in a periodic lattice in order to establish well-placed intermediate band boundaries. Current self-organized growth technology can achieve this requirement when certain criteria are satisfied, such as a lattice mismatch between the QD and barrier.

We proceeded to highlight potential QD-IBSC materials that have energetic optimized levels to achieve a theoretical efficiency greater than 70%. The design criteria used to search barrier/quantum dot material includes negligible valence band offsets between the barrier and quantum dot to prevent unnecessary bands from forming, direct band materials used to strengthen the absorption coefficient, and a lattice mismatch between the barrier/quantum dot material so self organized quantum dots could grow.

Our study indicates that efficiency is somewhat robust for the placement of these bands. It was determined that if the cell is designed to a slightly lower efficiency it could more easily accommodate variations in the intermediate band energies. A minimum distance between the quantum dots was determined to prevent the overlap of the intermediate bands with each other and valence/conduction bands.

Chapter 4: Finite Element Method and its Application to Schrödinger's Equation

In Chap. 3, we found QD-IBSC materials that boost thermodynamic efficiencies to over 70% with two intermediate bands. However, these materials were selected under certain restrictive assumptions that allowed for analytic solutions to the proposed model. This chapter and the next chapter will attempt to bridge the gap from the analytic solution to a more refined numerical solution, so the restrictive assumptions can be relaxed in order to search for a more realistic QD-IBSC. By doing so, we must diverge from the majority of the discussions thus far and switch our focus to the finite element method, which will become the bridge that allows for refined solutions. Our research and development of the finite method is done so with its application to the QD-IBSC in mind. As such, we will incorporate ideas of the previous chapters as necessary.

Analytic solutions to Schödinger's equation are few. Two of the more notable ones are the quantum oscillator potential and Coulomb potential, which Schrödinger solved in his first paper on quantum mechanics [41]. Other analytic solutions exist for simple potentials that include square or spherically symmetric potential wells. These solutions are useful in terms of our understanding of the atomic universe. However, analytic solutions are not available for practical applications, especially those that involve emerging technology. Today, the advent of computers and desire to understand typically intractable quantum systems drive numerical methods that turn Schödinger's equation into matrix mechanics.

The finite element method is one such method that transforms differential equations into matrix operations. The method originates in the principal of stationary action, which should resonate with physicists, and its strength lies at its ability to handle complicated boundary value problems. Applying FEM to Schödinger's equation is done in a straightforward manner that allows calculations in one, two, and three dimensions.

We begin in Sec. 4.1 with an overview of the method in the context of Schödinger's equation and apply it to the infinite potential well in one dimension. The next few sections, Sec. 4.2 - 4.4, continue to develop the method but with levels of sophistication required to find energy levels associated with the QD-IBSC. We conclude with Sec. 4.5.

4.1 Finite Element Method

The finite element method (FEM) has its origins in structure mechanics but was broadened to include other areas after it was generalized by Zienkiewicz and Cheung using variational procedures [42, 43]. Specifically, they interpreted FEM as finding a function minimizing (or maximizing) a certain specified functional over the whole field involved.

Newton's laws of motion for a particle experiencing forces, temperature distribution in a thermally conducting system, electrostatic potential in a region occupied by charges, the wavefunction of a quantum mechanical particle, etc. all obey differential equations that can be derived from the principal of stationary action[1] [44]. The principle of energy minimization and Hamilton's principle in classical mechanics are both manifestations of this single principle. There are two equal but different formulations to these types of problems: (1) attempt to directly solve the governing differential equation, or (2) evoke the principle of stationary action by casting it into a variational form. Due to the synthesis of the principle of stationary action and dynamic equations governing physical quantities, FEM is a natural extension to most branches in physics. Standard FEM references for material in this chapter are [45, 46, 47, 48, 49].

The application of this method normally takes place in two distinct steps. In the first step, we define the geometry of the problem and tessellate the region into simplices or elements. In the

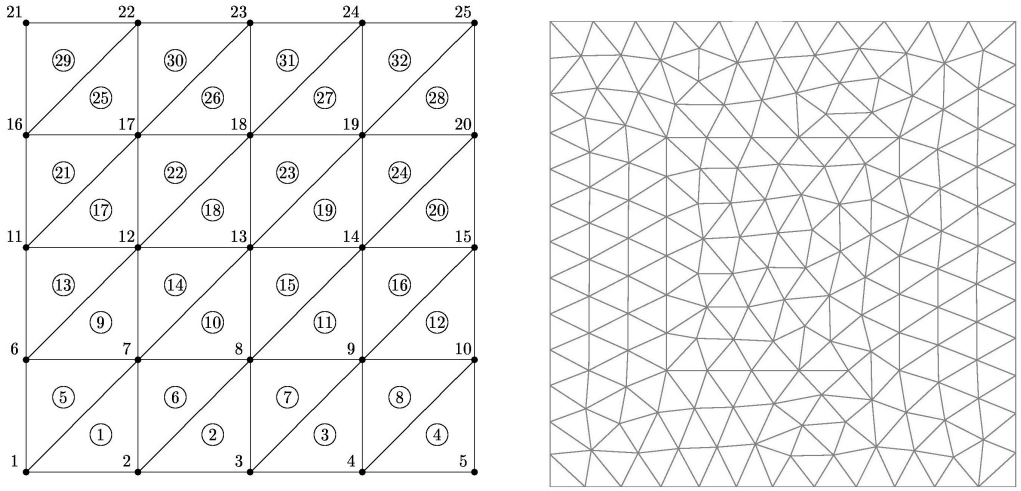
[1] This principle states that for a conservative system the action integral is stationary for the physical quantity.

second step, we reformulate the dynamics into a variational problem[2] and the geometry defined in step one is used to transform the variational problem into a matrix problem. Once the problem is in a matrix representation, we solve to find the energy eigenvalues and the corresponding eigenvectors. This is the general idea. We begin the discussion at step one by defining what is called a mesh.

The physical region is first discretized by drawing a boundary and then strategically placing points within the boundary. The adjacent points are then connected by lines. The adjacent lines are then connected to form areas. The adjacent areas are then connected to form volumes and this process continues up to the dimension of the physical region. The end result is called a mesh. Those smaller regions are called elements and the points are called nodes. The nodes and the elements are independently numbered. Lines in one dimension, areas in two dimensions, and volumes in three dimensions are all considered elements.

The geometrical shape of elements are polygons in 2D and polyhedra in 3D. The physical domain and user experience typically determine the type of element used. We will be using triangles in 2D and tetrahedra in 3D.

Within a mesh, all the elements must have the same geometric shape. However, the elements are not required to be 'structured' in anyway. By this we mean that the nodes, which are the vertices of elements, are not required to be regularly spaced throughout the mesh. A structured mesh (see Fig. 4.1a) is easy to generate but not practical of representing complicated regions of space. It is mostly used for illustrative purposes or as a first step in unstructured mesh code. On the other hand, a mesh that is generated when various inputs are considered is called unstructured (see Fig. 4.1b). For example, it is sometimes necessary to fix nodes or require more nodes near boundaries. The resulting mesh is unstructured. Generating this type can be difficult and normally involves complicated codes. A simple but adaptive unstructured mesh generator available free to the public called Distmesh[50] was used in generating Fig. 4.1b and all other FEM related problems throughout this thesis (see Chap. 5). It is a MATLAB based code that determines the node locations by solving for equilibrium in a truss structure (using piecewise linear force-displacement relations) and resets the elements by the Delaunay algorithm.



(a) **(b)**

Figure 4.1: (a) A structured mesh with triangular elements; (b) An unstructured mesh with triangular elements. In the structured mesh vertices are labeled by integers (1-25) and elements by circled integers (1-32).

[2]There are other approaches categorized as finite element methods that do not involve a variation. The method of weighted residuals is once such technique and includes a variety of different approaches within this category. However, the variational method is used throughout the text due to its intuitive nature to the types of problems encountered.

As mentioned, the mesh is the first step in solving a differential equation using finite elements. It defines the geometry of the problem, which can be very complicated. This alone makes the method advantageous. Further, consider nodes that are placed on a region that separates different materials. An internal boundary between the two materials is now defined and allows implementation of boundary conditions with relative ease. With respect to Schödinger's equation, the wavefunction and its derivative[3] are required to be continuous across the boundary. This is handled using finite elements as demonstrated in the later sections of this chapter.

Schödinger originally derived his equation from the action integral by placing a variation on function ψ such that any variation of it is stationary [41]. The result was the following[4]

$$\delta \int_\Omega \left(\frac{\hbar^2}{2m} (\nabla \psi^*(x))(\nabla \psi(x)) + \psi^*(x) V(x) \psi(x) - \psi^*(x) E \psi(x) \right) d\Omega = 0. \tag{4.1}$$

where the Lagrange multiplier, E, interpreted as energy, is used to enforce the normalizing condition $\int \psi^*(x)\psi(x) = 1$. The above equation is not normally expressed in its integral form but rather in its equivalent second order partial differential equation form. To see this, we carry out an integration by parts (divergence theorem) on the kinetic portion of the integral and treat $\psi^*(x)$ and $\psi(x)$ independently

$$\int_\Omega \frac{\hbar^2}{2} \frac{\nabla \psi^*(x) \cdot \nabla \psi(x)}{m(x)} d\Omega = \frac{\hbar^2}{2} \int_\Gamma \frac{\psi(x)^* \nabla \psi(x)}{m(x)} d\Gamma - \frac{\hbar^2}{2} \int_\Omega \psi(x)^* \nabla \cdot \left(\frac{\nabla \psi(x)}{m(x)} \right) d\Omega. \tag{4.2}$$

When the surface integral vanishes, the kinetic energy is replaced by the second term on the left hand side of Eq. 4.2 resulting in:

$$\delta \int_\Omega \psi^*(x) \left(-\frac{\hbar^2}{2} \nabla \cdot \left(\frac{\nabla \psi(x)}{m(x)} \right) + V(x)\psi(x) - E\psi(x) \right) d\Omega = 0. \tag{4.3}$$

For the integral to vanish, we recover Schödinger's equation as a second order differential equation

$$-\frac{\hbar^2}{2} \nabla \cdot \left(\frac{\nabla \psi(x)}{m(x)} \right) + V(x)\psi - E\psi(x) = 0. \tag{4.4}$$

In this derivation, we did not make the assumption that the mass was constant to emphasize the continuity conditions placed across boundaries. For a spatially varying mass, as in the effective mass approximation, the wavefunction and its inverse mass derivative should be continuous. However, when the mass is constant than 'normal' continuity conditions are recovered.

It is worth noting that the above process is entirely reversible and one can start with the differential equation to construct the action integral. To see this, consider Schödinger's differential equation 4.4 and multiply by a test function φ and integrate over all space.

$$\int_\Omega \left(\frac{-\hbar^2}{2m(x)} \nabla^2 \psi + V(x)\psi - E\psi \right) \varphi \, d\Omega = 0 \tag{4.5}$$

Using the divergence theorem, we obtain the following

$$\underbrace{-\frac{\hbar^2}{2} \int_\Gamma \frac{\varphi \nabla \psi}{m(x)} d\Gamma}_{=0} + \int_\Omega \left(\frac{\hbar^2}{2m(x)} (\nabla \varphi)(\nabla \psi) + \varphi V(x)\psi - \varphi E \psi \right) d\Omega = 0. \tag{4.6}$$

The surface term that arises from the integration will disappear when the wave function vanishes at infinity[5]. This will be the case, in general, when there are boundary conditions placed on the

[3] In the effective mass approximation, the wavefunction and the inverse mass derivative are required to be continuous across the boundary, see Eq. 4.4

[4] The steps needed to derive Schödinger's equation in this manner are detailed in Appendix A.

[5] In order to obtain the continuous spectrum, Schödinger placed the postulate that $\varphi(x) \to 0$ at infinity or at least

wavefunction. Energy eigenvalues will turn out to be discrete and the solution is termed bound. In the context of this discussion, bound states are the focus and the surface term is dropped accordingly, which results in:

$$\delta \left(\int_{\Omega} \frac{\hbar^2}{2m(x)} (\nabla\psi)^* (\nabla\psi) + \psi^* V(x) \psi - \psi^* E \psi \right) d\Omega = 0. \tag{4.7}$$

where we have set $\varphi = \psi^*$.

The above equation, Eq. 4.7, is in the form that can be exploited using mesh elements. Within each element, the wave function is approximated by a linear combination of a finite set of basis functions. These basis functions are defined by their values at each node and are zero outside the element. For illustration purposes, we show an example of one dimensional basis functions in Fig. 4.2. The two linear basis functions in element α are $\phi_j^\alpha(x) = 1 - x$ and $\phi_k^\alpha(x) = x$ within the range $[0, 1]$. At each nodal point ($x_j = 0$ and $x_k = 1$), the respective basis function is unity but drops to zero at all other nodal points.

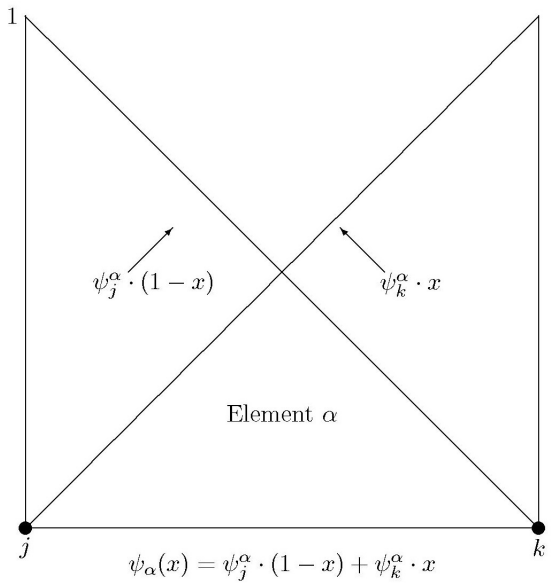


Figure 4.2: An example of a one dimensional element, α, with two nodes located at $j = 0$ and $k = 1$. Within the element, the wavefunction, $\psi(x)$, is represented by $\psi_\alpha(x)$ and is approximated by a linear combination of two interpolation basis functions. At node j, $\psi_\alpha(x_j) = \psi_j^\alpha$ and at node k, $\psi_\alpha(x_k) = \psi_k^\alpha$.

To be more explicit, if the number of elements is N_E and each elemental wave function is ψ_α

$$\psi(x) = \sum_{\alpha=1}^{N_E} \psi_\alpha(x). \tag{4.8}$$

By substituting Eq. 4.8 into the variational equation Eq. 4.7, each of the terms begins to simplify. The kinetic energy term is as follows:

$$\int_{\Omega} \frac{\hbar^2}{2m(x)} (\nabla\psi^*)(\nabla\psi) d\Omega = \int_{\Omega} \frac{\hbar^2}{2m(x)} \sum_{\beta=1}^{N_E} \nabla\psi_\beta(x)^* \sum_{\alpha=1}^{N_E} \nabla\psi_\alpha(x) d\Omega \Rightarrow \sum_{\alpha=1}^{N_E} \int_{\Omega_\alpha} \frac{\hbar^2}{2m(x)} \nabla\psi_\alpha^*(x) \nabla\psi_\alpha(x) d\Omega \tag{4.9}$$

The elemental wave function $\psi_\alpha(x)$ is zero outside its element α and similarly $\psi_\beta(x)$ is zero outside

it tends to a finite value at infinity. In the latter case, surface harmonics cause the integral to disappear.

its element β. Only when $\alpha = \beta$, the kinetic energy term is nonzero and this causes the double sum to be reduced to the single sum. The potential energy and energy eigenvalue terms are treated the same way:

$$\sum_{\alpha=1}^{N_E} \int_{\Omega_\alpha} \psi_\alpha^*(x) V(x) \psi_\alpha(x) d\Omega \tag{4.10}$$

$$E \sum_{\alpha=1}^{N_E} \int_{\Omega_\alpha} \psi_\alpha^*(x) \psi_\alpha(x) d\Omega \tag{4.11}$$

In principle, the three expressions above set up the finite element approximation. In order to see this, we introduce the elemental wave function as:

$$\psi_\alpha(x) = \sum_{i=1}^{n} \psi_i^\alpha \phi_i^\alpha(x) \tag{4.12}$$

where ψ_i^α are unknown coefficients that represent the amplitude of the wave function at a particular node[6], $\phi_i^\alpha(x)$ are basis functions, and the summation's upper limit is the number of basis functions per element. Inserting Eq. 4.12 into Eqs. 4.9-4.11 gives the following:

$$\begin{aligned}
&\sum_{\alpha=1}^{N_E} \sum_{i,j}^{n} \psi_i^{\alpha *} \underbrace{\left(\frac{\hbar^2}{2m^\alpha} \int_\Omega \left(\nabla \phi_i^\alpha(x)\right)^* \left(\nabla \phi_j^\alpha(x)\right) d\Omega \right)}_{KE_{ij}^\alpha} \psi_j^\alpha, \\
&\sum_{\alpha=1}^{N_E} \sum_{i,j}^{n} \psi_i^{\alpha *} \underbrace{\left(\int_\Omega \phi_i^{\alpha *}(x) V(x) \phi_j^\alpha(x) d\Omega \right)}_{PE_{ij}^\alpha} \psi_j^\alpha, \\
&E \cdot \sum_{\alpha=1}^{N_E} \sum_{i,j}^{n} \psi_i^{\alpha *} \underbrace{\left(\int_\Omega \phi_i^{\alpha *}(x) \phi_j^\alpha(x) d\Omega \right)}_{O_{ij}^\alpha} \psi_j^\alpha
\end{aligned} \tag{4.13}$$

The integrals are carried out and inserted into Eq. 4.7,

$$\sum_{\alpha=1}^{N_E} \sum_{i,j}^{n} \psi_i^{\alpha *} \left(KE_{ij}^\alpha + PE_{ij}^\alpha - E \cdot O_{ij}^\alpha \right) \psi_j^\alpha = 0, \tag{4.14}$$

resulting in a generalized eigenvalue equation with E representing the eigenvalues and ψ_i^α representing the eigenvectors. This equation represents a discretized version of action integral and we have to place a variation on the variable $\psi^{\alpha *}$ to exercise the principle of stationary action.

$$\frac{\delta A}{\delta \psi^{\alpha *}} = 0 \Rightarrow \sum_{\alpha=1}^{N_E} \sum_{i,j}^{n} \left(KE_{ij}^\alpha + PE_{ij}^\alpha - E \cdot O_{ij}^\alpha \right) \psi_j^\alpha \tag{4.15}$$

Once the boundary conditions are properly treated within the matrix, it is input into any standard generalized eigenvalue solver. Eigenvalues represent the energy and corresponding eigenvectors represent nodal wavefunction values.

To make the ideas of the previous paragraphs a bit more concrete, it is useful to demonstrate this approach on a standard quantum mechanical problem. This allows for the comparison and possible calibration of the approximate solution to the analytical solution. Lets consider a particle of mass, m, localized in a one dimensional infinite potential well of width L. Inside the well, $0 < x < L$, the potential is $V(x) = 0$ and outside the well, $x > L, x < 0$, $V(x) = \infty$. The bound energy states

[6]The coefficients might also represent the amplitude of the derivative of the wave function at a particular node.

and corresponding normalized wavefunction are found analytically by solving Schödinger's equation directly:

$$E_n = \frac{n^2\hbar^2\pi^2}{2mL^2}, \quad n = (1, 2, 3, \ldots)$$
$$\psi_n(x) = \sqrt{\frac{2}{L}}\sin(\frac{n\pi x}{L}), \quad n = (1, 2, 3, \ldots) \tag{4.16}$$

The finite element approximation begins by defining a mesh. This is done in almost the simplest way possible: 6 nodes defining 5 elements consisting of 2 nodes per element, each element has a length $L/5$, and the first node's physical position is $x_1 = 0$ and last node's physical position is $x_6 = L$ (see Fig. 4.3). Since there are 6 nodes, we are solving for 6 unknown wavefunction amplitudes and each matrix is 6×6 due to its represented quadratic form as in Eq. 4.14. The boundary conditions require the wavefunction to be 0 at nodes x_1 and x_6. To implement these conditions, the first row and column that represent ψ_1 are removed from each of the matrices. In addition, the last row and column that represent ψ_6 are removed from each of the matrices. Thus the matrices are reduced in size from 6×6 to 4×4.

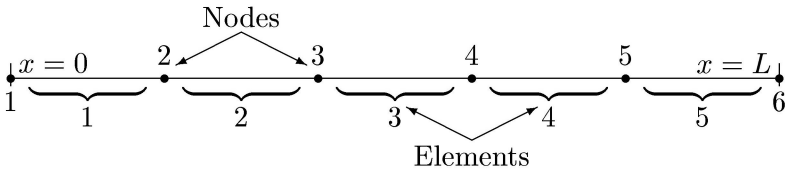


Figure 4.3: A five element one dimensional mesh.

Each elemental wavefunction is a linear combination of two basis functions:

$$\psi_\alpha(x) = \sum_{i=\alpha}^{\alpha+1} \psi_i^\alpha \phi_i^\alpha(x) \tag{4.17}$$

with the unknown wavefunction amplitudes ψ_i^α satisfying the requirement

$$\psi_{i+1}^\alpha = \psi_i^{\alpha+1} \tag{4.18}$$

to enforce the continuity of the wavefunction between elements. This allows us to drop the superscript α from ψ_i^α in Eq. 4.17. The basis functions ϕ_i^α in Eq. 4.17 are interpolation polynomials chosen so that the unknown wavefunction, $\psi(x)$, has the nodal wavefunction amplitude ψ_i at the node point x_i, i.e. $\psi(x_i) = \psi_i$ and drops linearly to zero at the adjacent nodes.

The kinetic energy, KE, and overlap, O, matrices are both symmetric but still involve a lot of integrals. A practical solution is to define a benchmark element in terms of local coordinates with local basis functions. The integrals are performed and the benchmark element is linearly mapped to each global element. Instead of performing integrals for every element located in the mesh, we only perform those associated with the benchmark element. This technique is developed in Sec. 4.2 and increases in sophistication when considering higher dimensions and degrees of freedom.

For this approximation, we are going define the benchmark element in the local coordinate system ξ with the two nodes located at -1 and 1. We use two linear basis functions that have the desired properties $\phi_1(-1) = 1$, $\phi_1(1) = 0$, $\phi_2(-1) = 0$, and $\phi_2(1) = 1$ in order to satisfy the conditions placed on the nodal wavefunction amplitudes. The resulting functions are

$$\phi_1(\xi) = \frac{1-\xi}{2}, \quad \phi_2(\xi) = \frac{1+\xi}{2} \quad -1 \le \xi \le 1 \tag{4.19}$$

and the elemental wavefunction originally represented by Eq. 4.17 has now transformed into

$$\psi_\alpha(x) = \sum_{i=1}^{2} \psi_{(\alpha-1+i)}\phi_i(\xi) \tag{4.20}$$

The basis functions can also used as the coordinate transformation. This is done by setting

$$x = \sum_{i=1}^{2} x_i\phi_i(\xi) \tag{4.21}$$

where x_i is the location of the two nodes from element α. This coordinate transformation and basis functions allow for the calculation of all matrix elements. Consider element $\alpha = 1$, the elemental wavefunction is $\psi_1(x) = \psi_1\phi_1(\xi) + \psi_2\phi_2(\xi)$. The kinetic energy matrix consists of the the inner product of the gradient

$$\left(\frac{d\psi_1}{d\xi}\frac{d\xi}{dx}\right)^* \cdot \left(\frac{d\psi_1}{d\xi}\frac{d\xi}{dx}\right) = \frac{1}{(x_2-x_1)^2} \times [\psi_1^* \quad \psi_2^*] \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} \tag{4.22}$$

which is than cast into the integral

$$\frac{\hbar^2}{2m}\frac{1}{(x_2-x_1)^2}\int_{-1}^{1} [\psi_1^* \quad \psi_2^*] \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} \frac{dx}{d\xi}d\xi = \frac{\hbar^2}{2m}\frac{5}{L}[\psi_1^* \quad \psi_2^*] \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} \tag{4.23}$$

We have replaced $x_2 - x_1$ by the length of the element, which is $L/5$. The overlap matrix is the inner product of the wavefunction cast into the integral

$$E \cdot \int_{-1}^{1} [\psi_1^* \quad \psi_2^*] \begin{bmatrix} \phi_1^*\phi_1 & \phi_1^*\phi_2 \\ \phi_2^*\phi_1 & \phi_2^*\phi_2 \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} \frac{dx}{d\xi}d\xi = \frac{E}{6}\frac{L}{5} \cdot [\psi_1^* \quad \psi_2^*] \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix}. \tag{4.24}$$

All the elemental matrices are solved for and carefully 'assembled' into global matricies which represent the generalized eigenvalue problem.

$$[\psi_1 \quad \psi_2 \quad \psi_3 \quad \psi_4 \quad \psi_5 \quad \psi_6] \left\{ \frac{\hbar^2}{2m}\frac{5}{L} \cdot \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 1+1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1+1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1+1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1+1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} - \frac{E}{6}\frac{L}{5} \cdot \begin{bmatrix} 2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2+2 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2+2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2+2 & 1 & 0 \\ 0 & 0 & 0 & 1 & 2+2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 2 \end{bmatrix} \right\} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \\ \psi_6 \end{bmatrix} = 0 \tag{4.25}$$

The addition in the diagonals is used to emphasis that the global 6×6 matrices are assembled from the local 2×2 matrices. Finally, we apply the boundary condition that the wavefunction must be zero at the ends, $\psi(0) = 0$ and $\psi(L) = 0$. This requires $\psi_1 = \psi_6 = 0$, thus each of the matrices have their first and last column terms are removed. In addition, the terms in the first and last row are removed which enforces $\psi_1^* = \psi_6^* = 0$.

Most technical computing software has built in algorithms that solve the generalized eigenvalue problem. Maple and MATLAB are two of the more popular commercial packages that include this

feature [51, 52]. In this example, Maple was utilized to find the lower energy levels and corresponding wavefunction by solving Eq. 4.25 after applying the boundary conditions. Table 4.1 compares the first four energies obtained from the approximation using 5, 20, and 100 elements to the exact solution. In all three approximations, the lower energy levels had a better convergence toward the

Table 4.1: The first four energy eigenvalues of the infinite potential well using FEM approximation with linear basis functions. The approximation was carried out using 5, 20, and 100 elements.

Quantum number	Energy eignvalues $(\hbar^2\pi^2 \ / \ 2mL^2)$			Exact eigenvalue $(\hbar^2\pi^2 \ / \ 2mL^2)$
	5 elements	20 elements	100 elements	
1	1.03331	1.00205	1.00008	1
2	4.54812	4.03300	4.00132	4
3	11.76515	9.16775	9.00666	9
4	23.08493	16.53300	16.02106	16

exact energy. As the quantum number increases, the 5 element approximation quickly diverges as opposed to the 20 and 100 element approximation. There are two important generalizations observed from this simple calculation: the approximate energy values are most accurate with the lower energy levels associated with a quantum system and as the number of elements increases.

The wavefunction using a 20 element finite element approximation was determined and superimposed on the exact solution to the two lowest eigenvalues, Eq. 4.16 (see Fig. 4.4). The eigenvectors, which represent the wavefunction amplitudes at each node, are used to approximate the wavefunction. As seen, there are negligible differences between the exact solution and approximation.

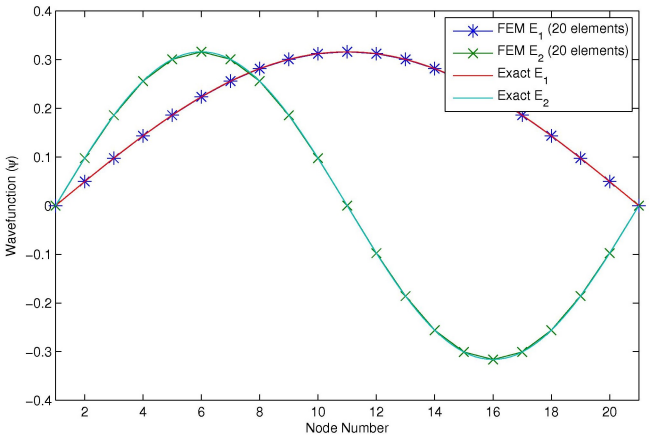


Figure 4.4: A 20 element finite element approximation to the two lowest eigenstates.

The eigenvectors are not orthonormal with each other due to the non-orthogonality of the basis functions. However, they are orthonormal with respect to the overlap matrix.

$$\psi^*_{i,\alpha} O_{ij} \psi_{j,\beta} = \delta_{\alpha\beta} \tag{4.26}$$

where matrix ψ's columns refer to the eigenvectors of the corresponding eigenvalue, i.e. wavefunction amplitude at each node. Here α, β index eigenvectors not elements. Equation 4.26 represents the

condition that bound states wavefunctions, $\Psi_i(x)$, are subject to the normalization condition:

$$\int_{-\infty}^{\infty} |\Psi_i(x)|^2 dx = 1 \tag{4.27}$$

Further, the eigenvectors also display the following property with the respect to the total energy, $KE + PE$,

$$\psi^*_{i,\alpha} \left(KE_{ij} + PE_{ij}\right) \psi_{j,\beta} = \delta_{\alpha\beta} E(\alpha). \tag{4.28}$$

It should be noted that the wavefunction can be approximated by the matrix eigenvectors **only** for a sufficient number of elements in a mesh. Otherwise, eigenvectors should be used in conjunction with basis functions in order to determine the wavefunction within each element. We recommend using the latter version of the wavefunction for applications involving quantitative calculations and the former for simple qualitative applications, i.e. plotting. As a quantitative example, suppose that a particle is known to be in a certain bound state, say 1 with wavefunction Ψ_1. Then the probability that this particle will be found in a certain region of space, say Ω, will be

$$P = \int_{\Omega} |\Psi_1|^2 d\Omega \tag{4.29}$$

If N_Ω is a set of elements that only fall within the region Ω, then we can transform Eq. 4.29 into

$$P = \sum_{\alpha=1}^{N_\Omega} \int_{\Omega_\alpha} |\psi_\alpha|^2 d\Omega \tag{4.30}$$

where the summation on the right hand side only includes the elements falling within Ω and the integral only includes their representative elemental wavefunctions ψ_α. Equation 4.30 is similar to Eq. 4.11 except it includes contributions from the relevant elemental wavefunctions. In the same exact manner as the overlap matrix O_{ij} in Eq. 4.14, a *probability* matrix P_{ij} is developed. Then the probability that a particle is in region Ω for bound state 1 would just be

$$P = \psi^*_{i,1} P_{ij} \psi_{j,1} \tag{4.31}$$

where we have multiplied the column eigenvectors on the right and the row eigenvectors on the left.

4.2 Developing Basis Functions

In FEM, the action integral is discretized and the unknown function, $\psi(x)$ in our case, is solved for using a linear combination of elemental functions. These elemental functions are a linear combination of basis functions and chosen such that the unknown function has the value ψ_i at node i, i.e. $\psi(x_i) = \psi_i$. Further, these basis functions must ensure nodal continuity between adjacent elements. If it is desired that only $\psi(x)$ be continuous between elements than this is C_0 continuous. However, in some cases it is desired that both $\psi(x)$ and its derivative (or inverse mass derivative) be continuous between elements. This is C_1 continuous. The type of continuity conditions determine the basis functions. Once known, basis functions allow the integrals to be carried out.

The focus of this section is to develop the basis functions in one dimension for C_0 continuity and determine the elemental kinetic, potential, and overlap matrices required to solve Schödinger's equation in the FEM approximation. After setting up the basic ideas, we will then extend to solve Schödinger's equation in two and three dimensions.

To complete the discretization in FEM, we choose a basis that are comprised of piecewise polynomial functions, $\phi(x)$, and use this basis to interpolate the wavefunction. Within each element, basis functions are determined by the constraints placed on each nodal degree of freedom. These constraints must satisfy the requirement

$$\phi_i(x_j) = \delta_{ij} \tag{4.32}$$

in order for $\psi(x_i) = \psi_i$. Here, δ_{ij} equals 1 when $i = j$ and 0 when $i \neq j$. Let us first consider the line element that ranges between the interval $[x_1, x_2]$ and has two nodes, each located on the interval's end points. With C_0 continuity, there is one degree of freedom per node. Each degree of freedom requires a basis function to satisfy Eq. 4.32. Basis functions assume the form of an interpolation polynomial chosen up to the degree necessary to define each one. Therefore, there will be two basis functions and the polynomial for the element at hand will be linear:

$$\phi_i = a_i + b_i x \quad i = 1, 2 \tag{4.33}$$

where a_i and b_i are coefficients that are determined by the simultaneous equations set up by Eq. 4.32.

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \cdot \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.34}$$

The row entries of the matrix on the left refer to the values at nodal point x_i, the matrix on the right contains the unknown coefficients, and the matrix on the right hand side of Eq. 4.34 is just the identity matrix which imposes the condition Eq. 4.32. A matrix inversion determines the coefficients in each basis function.

$$\begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix}^{-1} \Rightarrow \begin{bmatrix} -\frac{x_2}{-x_2+x_1} & \frac{x_1}{-x_2+x_1} \\ \frac{1}{-x_2+x_1} & -\frac{1}{-x_2+x_1} \end{bmatrix} \tag{4.35}$$

$$\begin{aligned} \phi_1(x) &= -\frac{x_2}{-x_2+x_1} + \frac{x}{-x_2+x_1} = \frac{x-x_2}{x_1-x_2} \\ \phi_2(x) &= \frac{x_1}{-x_2+x_1} - \frac{x}{-x_2+x_1} = \frac{x_1-x}{x_1-x_2} \end{aligned} \tag{4.36}$$

The above procedure can be extended to include additional nodes within an element. For example, consider a line element defined to include 3 nodes located at x_1, x_2, and x_3 within the interval $[x_1, x_3]$. Three nodes require three basis functions and the interpolation polynomials are cubic to satisfy Eq. 4.32. The matrix equations are set up, as in Eq. 4.34, and the coefficients are determined through a matrix inversion.

It is beneficial in FEM to define a benchmark element and linearly map all elements in the mesh to the benchmark element. This way only one element is defined and all the integrals can be carried out since the Jacobian will be constant. We reserve ξ for the benchmark or local coordinate and x for the global coordinate.

The benchmark line element used to construct the elemental kinetic, potential, and overlap matrix is defined by two nodes located at $\xi_1 = -1$ and $\xi_2 = 1$ within the $[-1, 1]$ interval. The basis functions are determined by Eq. 4.36 and are:

$$\begin{aligned} \phi_1(\xi) &= \frac{1-\xi}{2} \\ \phi_2(\xi) &= \frac{1+\xi}{2} \end{aligned} \tag{4.37}$$

Therefore, each elemental wavefunction will consist of a linear combination of these two basis functions. To define the elemental matrices, we consider the first element $\alpha = 1$ in a mesh that contains nodes $n = 1, 2$ so that the elemental wavefunction is:

$$\Psi_1(x) \rightarrow \psi_1\phi_1(\xi) + \psi_2\phi_2(\xi) \tag{4.38}$$

where ψ_i is the unknown wavefunction amplitude at node i.

4.2.1 Elemental Kinetic Matrix In 1D

The elemental kinetic matrix is constructed from:

$$\frac{\hbar^2}{2m}\int(\nabla_x\Psi_1(x))^2dx = \frac{\hbar^2}{2m}\int_{-1}^{1}(\partial_\xi\Psi_1(\xi)\cdot\partial_x\xi)^2\frac{dx}{d\xi}d\xi \tag{4.39}$$

where ∂_i is the partial derivative with respect to i. We have exploited the change of variables $x \to \xi$ using the chain rule in the above equation. Conveniently, the global coordinate x itself can be represented in terms of the basis functions:

$$x = x_1\phi_1(\xi) + x_2\phi_2(\xi) \to x_1\frac{1-\xi}{2} + x_2\frac{1+\xi}{2} \tag{4.40}$$

$$\partial_\xi x = \frac{x_2 - x_1}{2} \tag{4.41}$$

$$\partial_x\xi = \frac{2}{x_2 - x_1} \tag{4.42}$$

Since the change of variables between x and ξ is linear, the terms $\partial_x\xi$ and $\frac{dx}{d\xi}$ are pulled outside the integral.

$$\frac{\hbar^2}{2m}\cdot(\partial_x\xi)^2\cdot\frac{dx}{d\xi}\int_{-1}^{1}(\partial_\xi\Psi_1(\xi))^2d\xi \tag{4.43}$$

The integrals are carried out and the quadratic form of the elemental wavefunction represented in the kinetic matrix is:

$$\frac{\hbar^2}{2m}\cdot(\partial_x\xi)^2\cdot\frac{dx}{d\xi}\times\begin{bmatrix}\psi_1^* & \psi_2^*\end{bmatrix}\begin{bmatrix}1/2 & -1/2\\ -1/2 & 1/2\end{bmatrix}\begin{bmatrix}\psi_1\\ \psi_2\end{bmatrix} \tag{4.44}$$

The kinetic energy matrix is symmetric as expected and it only remains to place each elemental matrix entry in the global $N \times N$ kinetic matrix for an N node mesh.

4.2.2 Elemental Potential Matrix In 1D

The elemental potential matrix is constructed from:

$$\int\Psi_1^*(x)V(x)\Psi_1(x)dx \tag{4.45}$$

where the potential energy $V(x)$ is included in the integral. Under the change of variables as in the kinetic energy matrix, $x \to \xi$, the wavefunction becomes $\Psi_1^*(x) \to \Psi_1^*(\xi)$ along with the potential energy $V(x) \to V(\xi)$. However, a piecewise linear approximation to the wavefunction has already taken place (see Eq. 4.38) and, as such, a similar piecewise linear approximation to the potential energy would be appropriate. If V_1 and V_2 are the values of the potential located at the nodes, then the potential energy and integral Eq. 4.45 have transformed to the following:

$$V(x) \to V_1\phi_1(\xi) + V_2\phi_2(\xi) \tag{4.46}$$

$$\int_{-1}^{1}(\psi_1\phi_1(\xi) + \psi_2\phi_2(\xi))^2(V_1\phi_1(\xi) + V_2\phi_2(\xi))\frac{dx}{d\xi}d\xi \tag{4.47}$$

where $\frac{dx}{d\xi}$ can be pulled outside of the integral due to the linear change of variables. The integral in Eq. 4.47 represents a trilinear product of the basis functions and a bilinear product of the two unknowns ψ_i. Integrals are carried out and the quadratic form of the elemental wavefunction represented in the potential matrix is:

$$\frac{dx}{d\xi}\times\begin{bmatrix}\psi_1^* & \psi_2^*\end{bmatrix}\begin{bmatrix}1/2\,V_1 + 1/6\,V_2 & 1/6\,V_1 + 1/6\,V_2\\ 1/6\,V_1 + 1/6\,V_2 & 1/6\,V_1 + 1/2\,V_2\end{bmatrix}\begin{bmatrix}\psi_1\\ \psi_2\end{bmatrix} \tag{4.48}$$

Each matrix entry contains both V_1 and V_2 as a weighted average, a more sophisticated approach than assuming an average value for potential over the entire element. Similar to the kinetic energy matrix, the potential energy matrix is also symmetric. It only remains to assemble each elemental matrix entry in the global $N \times N$ potential energy matrix for an N node mesh. If the potential energy values $V_1 = V_2$ in a element, the matrix is proportional to the overlap matrix (see Eq. 4.50). This is an expected and necessary result.

4.2.3 Elemental Overlap Matrix In 1D

The elemental overlap matrix is constructed from:

$$\int \Psi_1^*(x)\Psi_1(x)dx \rightarrow \int_{-1}^{1} (\psi_1\phi_1(\xi) + \psi_2\phi_2(\xi))^2 \frac{dx}{d\xi} d\xi \tag{4.49}$$

where again we have exploited the linear change in variables and $\frac{dx}{d\xi}$ can be pulled outside the integral. The integral on the right hand side represents a bilinear product of the basis functions and a bilinear product of the two unknowns ψ_i. The integral is carried out and the quadratic form of the elemental wavefunction represented in the elemental overlap matrix is:

$$\frac{dx}{d\xi} \times [\psi_1^* \quad \psi_2^*] \begin{bmatrix} 2/3 & 1/3 \\ 1/3 & 2/3 \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} \tag{4.50}$$

Similar to the kinetic and potential energy matrix, the overlap energy matrix is also symmetric. It only remains to assemble each elemental matrix entry in the global $N \times N$ overlap energy matrix for an N node mesh.

4.3 Higher Dimensions

An extension of FEM from one dimension to higher dimensions is natural in terms of the ideas already presented. However, the complexity involved in construction the elemental matrices is greater. In this section, we first define the basis functions in two and three dimensions for the benchmark triangle and tetrahedron respectively and then proceed to construct each of the elemental matrices.

The benchmark triangle is referred to as the standard triangle defined in the local coordinates (ξ, η) with vertices located at $(0, 0)$, $(1, 0)$, and $(0, 1)$. Each triangular element in an FEM mesh has its vertices located at (x_i, y_i) $i = 1, 2, 3$ and is linearly mapped to the benchmark element. As such, the basis functions for the standard triangle are defined in local coordinates. Similar to 1D basis functions, 2D basis functions must satisfy a modified version of Eq. 4.32 or $\phi_i(\xi_j, \eta_j) = \delta_{ij}$.

We assume the basis functions have a linear polynomial form and each degree of freedom requires a basis function. Therefore, there are 3 basis functions with 9 constants (a_i, b_i, c_i) to be determined:

$$\phi_i = a_i + b_i\xi + c_i\eta \quad i = 1, 2, 3 \tag{4.51}$$

The basis functions are constructed from setting up the simultaneous equations in matrix form as before.

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.52}$$

A matrix inversion determines (a_i, b_i, c_i) and forms the basis functions:

$$\begin{aligned} \phi_1(\xi, \eta) &= 1 - \xi - \eta \\ \phi_2(\xi, \eta) &= \xi \\ \phi_3(\xi, \eta) &= \eta \end{aligned} \tag{4.53}$$

Similar to 1D, triangular elemental wavefunctions $\Psi_\alpha(x, y)$ and the coordinate transformation $(\xi, \eta) \rightarrow$

(x, y) can be expressed as a linear combination of the basis functions:

$$\Psi_\alpha(x, y) = \sum_{i=1}^{3} \psi_i \phi_i(\xi, \eta) \tag{4.54}$$

$$x = \sum_{i=1}^{3} x_i \phi_i(\xi, \eta)\,,\; y = \sum_{i=1}^{3} y_i \phi_i(\xi, \eta) \tag{4.55}$$

where ψ_i is the unknown wavefunction amplitude at node i and (x_i, y_i) are the element's nodal positions.

In 3D, we assume the benchmark element is the standard tetrahedron and is defined in local coordinates (ξ, η, ζ) with vertices located at $(0, 0, 0)$, $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$. Each tetrahedral element in an FEM mesh has its vertices defined in global coordinates located at (x_i, y_i, z_i) $i = 1, 2, 3, 4$ and is linearly mapped to the benchmark element. As such, the basis functions for the standard tetrahedron are defined in local coordinates. Similar to 1D and 2D basis functions, 3D basis functions must satisfy a modified version of Eq. 4.32 or $\phi_i(\xi_j, \eta_j, \zeta_j) = \delta_{ij}$.

We assume the basis functions have a linear polynomial form and each degree of freedom requires a basis function. Therefore, there are 4 basis functions with 12 constants (a_i, b_i, c_i, d_i) to be determined:

$$\phi_i = a_i + b_i\xi + c_i\eta + d_i\zeta \quad i = 1, 2, 3, 4 \tag{4.56}$$

The basis functions are constructed by setting up the simultaneous equations in matrix form, as before, and carrying out a matrix inversion. The resulting basis functions are:

$$\begin{aligned} \phi_1(\xi, \eta, \zeta) &= 1 - \xi - \eta - \zeta \\ \phi_2(\xi, \eta, \zeta) &= \xi \\ \phi_3(\xi, \eta, \zeta) &= \eta \\ \phi_4(\xi, \eta, \zeta) &= \zeta \end{aligned} \tag{4.57}$$

Similar to 1D and 2D, tetrahedral elemental wavefunctions $\Psi_\alpha(x, y, z)$ and the coordinate transformation $(\xi, \eta, \zeta) \to (x, y, z)$ can be expressed as a linear combination of the basis functions:

$$\Psi_\alpha(x, y, z) = \sum_{i=1}^{4} \psi_i \phi_i(\xi, \eta, \zeta) \tag{4.58}$$

$$x = \sum_{i=1}^{4} x_i \phi_i(\xi, \eta, \zeta)\,,\; y = \sum_{i=1}^{4} y_i \phi_i(\xi, \eta, \zeta)\,,\; z = \sum_{i=1}^{4} z_i \phi_i(\xi, \eta, \zeta) \tag{4.59}$$

where ψ_i is the unknown wavefunction amplitude at node i and (x_i, y_i, z_i) are the element's nodal positions. To define the elemental matrices in two and three dimensions, we consider element $\alpha = 1$ and assume it contains nodes $n = 1, 2, 3$ or $n = 1, 2, 3, 4$ for the triangle and tetrahedron respectively.

4.3.1 Elemental Kinetic Matrix - Higher Dimensions

We begin by examining the two dimensional version of the kinetic elemental matrix. It is constructed from a similar form of Eq. 4.39 or

$$\frac{\hbar^2}{2m} \int\int (\nabla \Psi_1(x, y))^2 dx\, dy \to \frac{\hbar^2}{2m} \int\int \left(\frac{\partial \Psi_1(x, y)}{\partial x} \right)^2 + \left(\frac{\partial \Psi_1(x, y)}{\partial y} \right)^2 dx\, dy \tag{4.60}$$

The partial derivatives of the wavefunction are carried out using the chain rule via the 2×2 inverse Jacobian, $\mathbf{J^{-1}}$, for the coordinate transformation. A column vector containing the partial derivatives of the elemental wavefunction with respect to the global coordinates is represented by the inverse Jacobian matrix and partial derivatives of the elemental wavefunction with respect to the local

coordinates.

$$\begin{bmatrix} \frac{\partial \Psi_1(x,y)}{\partial x} \\ \frac{\partial \Psi_1(x,y)}{\partial y} \end{bmatrix} = \mathbf{J^{-1}} \begin{bmatrix} \frac{\partial \Psi_1(x,y)}{\partial \xi} \\ \frac{\partial \Psi_1(x,y)}{\partial \eta} \end{bmatrix} \rightarrow \mathbf{J^{-1}} \begin{bmatrix} \frac{\partial \phi_1(\xi,\eta)}{\partial \xi} & \frac{\partial \phi_2(\xi,\eta)}{\partial \xi} & \frac{\partial \phi_3(\xi,\eta)}{\partial \xi} \\ \frac{\partial \phi_1(\xi,\eta)}{\partial \eta} & \frac{\partial \phi_2(\xi,\eta)}{\partial \eta} & \frac{\partial \phi_3(\xi,\eta)}{\partial \eta} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \end{bmatrix} \tag{4.61}$$

On the right hand side of Eq. 4.61 we have made use of the benchmark elemental wavefunction from Eq. 4.55 and decomposed it into a 2×3 partial derivative matrix $\mathbf{D}$ and column vector containing the wavefunction amplitudes. The integrand of Eq. 4.60 is the inner product of the wavefunction derivative with itself:

$$\begin{bmatrix} \frac{\partial \Psi_1(x,y)}{\partial x} & \frac{\partial \Psi_1(x,y)}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial \Psi_1(x,y)}{\partial x} \\ \frac{\partial \Psi_1(x,y)}{\partial y} \end{bmatrix} = [\psi_1 \quad \psi_2 \quad \psi_3] \, \mathbf{D^t} \cdot \left(\mathbf{J^{-1}}\right)^{\mathbf{t}} \cdot \mathbf{J^{-1}} \cdot \mathbf{D} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \end{bmatrix} \tag{4.62}$$

We can further simplify the right hand side of the equation realizing $\left(\mathbf{J^{-1}}\right)^{\mathbf{t}} \cdot \mathbf{J^{-1}} \rightarrow \left(\mathbf{J} \cdot \mathbf{J^t}\right)^{\mathbf{-1}}$.

Transforming the integral of Eq. 4.60 into local coordinates, which requires the Jacobian determinant $|J|$, will represent the quadratic form of the kinetic energy matrix:

$$[\psi_1 \quad \psi_2 \quad \psi_3] \int_0^1 \int_0^{1-\eta} \mathbf{D^t} \cdot \left(\mathbf{J} \cdot \mathbf{J^t}\right)^{\mathbf{-1}} \cdot \mathbf{D} \, |J| \, d\xi \, d\eta \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \end{bmatrix} \tag{4.63}$$

with the limits of integration over the standard triangle.

One of the advantages of defining a benchmark element is to carry out the integrals only once. If this were not the case, the FEM procedure demands solving integrals for every element within a mesh through whatever means necessary. Numerical integration techniques are employed, typically Gaussian quadrature, potentially introducing error and increasing numerical operations. In Eq. 4.63, it is observed that $\left(\mathbf{J} \cdot \mathbf{J^t}\right)^{\mathbf{-1}}$ can not simply be brought outside of the integral even though its entries are constant, matrix multiplication is not commutative. This makes carrying out the integrals for Eq. 4.63 in the present form cumbersome and unappealing. A more sophisticated approach is to represent the quadratic form of the kinetic energy matrix through a trace operation involving a matrix $\mathbf{P}$, defined below, multiplied by $\left(\mathbf{J} \cdot \mathbf{J^t}\right)^{\mathbf{-1}}$.

We first define four 3×3 matrices $\mathbf{M}_{\mu,\nu}$, with $\mu = (\xi, \eta)$ and $\nu = (\xi, \eta)$, as integrals consisting of a product of basis function derivatives with respect to local coordinates ξ, η:

$$\mathbf{M}_{\mu,\nu}(i,j) = \int_0^1 \int_0^{1-\eta} \nabla_\mu \phi_i(\xi,\eta) \cdot \nabla_\nu \phi_j(\xi,\eta) \, d\xi \, d\eta \quad 1 \leq i,j \leq 3 \tag{4.64}$$

Each 2×2 matrix $\mathbf{P_{ij}}$ consists of a combination of the four matrices $\mathbf{M}_{\mu,\nu}$ with its entries

$$\mathbf{P_{ij}} = \begin{bmatrix} \mathbf{M}_{\xi,\xi}(i,j) & \mathbf{M}_{\xi,\eta}(i,j) \\ \mathbf{M}_{\eta,\xi}(i,j) & \mathbf{M}_{\eta,\eta}(i,j) \end{bmatrix} \quad 1 \leq i,j \leq 3 \tag{4.65}$$

Finally, the quadratic form of the kinetic energy matrix is built up from the trace

$$KE_{ij} = \frac{\hbar^2}{2m} |J| \psi_i \mathrm{Tr} \left(\mathbf{P_{ij}} \cdot \left(\mathbf{J} \cdot \mathbf{J^t}\right)^{\mathbf{-1}} \right) \psi_j \tag{4.66}$$

and is equivalent to Eq. 4.63. In this form, the terms involving the Jacobian matrix are brought outside of the integral. Each matrix $\mathbf{M}_{\mu,\nu}$ in Eq. 4.64 is solved for

$$\mathbf{M}_{\xi,\xi} = \begin{bmatrix} 1/2 & -1/2 & 0 \\ -1/2 & 1/2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{M}_{\xi,\eta} = \begin{bmatrix} 1/2 & 0 & -1/2 \\ -1/2 & 0 & 1/2 \\ 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{M}_{\eta,\xi} = \mathbf{M}^{\mathbf{t}}_{\xi,\eta}\,,\ \mathbf{M}_{\eta,\eta} = \begin{bmatrix} 1/2 & 0 & -1/2 \\ 0 & 0 & 0 \\ -1/2 & 0 & 1/2 \end{bmatrix} \tag{4.67}$$

and the elemental matrices are constructed. Now it only remains to assemble each elemental matrix entry in the global $N \times N$ kinetic energy matrix for an N node mesh.

In 3D, the kinetic energy matrix is built up exactly the same way as in 2D. The three dimensional version of the kinetic energy matrix is the following

$$\frac{\hbar^2}{2m}\int\int\int \left(\frac{\partial \Psi_1(x,y,z)}{\partial x}\right)^2 + \left(\frac{\partial \Psi_1(x,y,z)}{\partial y}\right)^2 + \left(\frac{\partial \Psi_1(x,y,z)}{\partial z}\right)^2 dx\,dy\,dz \tag{4.68}$$

and we utilize the 3×3 inverse Jacobian, $\mathbf{J^{-1}}$, for the coordinate transformation. The column vector containing the partial derivatives of the elemental wavefunction with respect to the global coordinates as in Eq. 4.61 is represented in the three dimensional form.

$$\begin{bmatrix} \frac{\partial \Psi_1(x,y,z)}{\partial x} \\ \frac{\partial \Psi_1(x,y,z)}{\partial y} \\ \frac{\partial \Psi_1(x,y,z)}{\partial z} \end{bmatrix} = \mathbf{J^{-1}} \begin{bmatrix} \frac{\partial \phi_1(\xi,\eta,\zeta)}{\partial \xi} & \frac{\partial \phi_2(\xi,\eta,\zeta)}{\partial \xi} & \frac{\partial \phi_3(\xi,\eta,\zeta)}{\partial \xi} & \frac{\partial \phi_4(\xi,\eta,\zeta)}{\partial \xi} \\ \frac{\partial \phi_1(\xi,\eta,\zeta)}{\partial \eta} & \frac{\partial \phi_2(\xi,\eta,\zeta)}{\partial \eta} & \frac{\partial \phi_3(\xi,\eta,\zeta)}{\partial \eta} & \frac{\partial \phi_4(\xi,\eta,\zeta)}{\partial \eta} \\ \frac{\partial \phi_1(\xi,\eta,\zeta)}{\partial \zeta} & \frac{\partial \phi_2(\xi,\eta,\zeta)}{\partial \zeta} & \frac{\partial \phi_3(\xi,\eta,\zeta)}{\partial \zeta} & \frac{\partial \phi_4(\xi,\eta,\zeta)}{\partial \zeta} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{bmatrix} \tag{4.69}$$

On the right hand side of Eq. 4.69 we have made use of the benchmark elemental wavefunction from Eq. 4.59 and decomposed it into the 3×4 partial derivate matrix $\mathbf{D}$ and column vector containing the wavefunction amplitudes. The integrand of Eq. 4.68 is the inner product of the wavefunction derivative with itself resulting in

$$[\psi_1 \quad \psi_2 \quad \psi_3 \quad \psi_4]\ \mathbf{D^t} \cdot \left(\mathbf{J} \cdot \mathbf{J^t}\right)^{\mathbf{-1}} \cdot \mathbf{D} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{bmatrix} \tag{4.70}$$

where we have made use of the simplification $\left(\mathbf{J^{-1}}\right)^{\mathbf{t}} \cdot \mathbf{J^{-1}} \rightarrow \left(\mathbf{J} \cdot \mathbf{J^t}\right)^{\mathbf{-1}}$. This result along with the Jacobian determinant, J, is used to transform Eq. 4.68 into the local coordinates and will represent the quadratic form of the kinetic energy matrix:

$$[\psi_1 \quad \psi_2 \quad \psi_3 \quad \psi_4] \int_0^1 \int_0^{1-\eta} \int_0^{1-\eta-\zeta} \mathbf{D^t} \cdot \left(\mathbf{J} \cdot \mathbf{J^t}\right)^{\mathbf{-1}} \cdot \mathbf{D}\,|J|\,d\xi\,d\eta\,d\zeta \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{bmatrix} \tag{4.71}$$

with the limits of integration over the standard tetrahedron. A more sophisticated form of elemental kinetic energy matrix can be constructed through the same means as in the two dimensional case. In fact, the resulting equation is exactly the same as Eq. 4.66. Differences emerge as we consider matrices $\mathbf{M}_{\mu,\nu}$ and $\mathbf{P}$ in the context of this additional dimension. The extra dimension requires $\mu = (\xi, \eta, \zeta)$ and $\nu = (\xi, \eta, \zeta)$, so there are nine 4×4 matrices $\mathbf{M}_{\mu,\nu}$ instead of four 3×3 matrices. Again, they are defined as integrals consisting of a product of basis function derivatives with respect to local coordinates ξ, η, ζ:

$$\mathbf{M}_{\mu,\nu}(i,j) = \int_0^1 \int_0^{1-\eta} \int_0^{1-\eta-\zeta} \nabla_\mu \phi_i(\xi,\eta) \cdot \nabla_\nu \phi_j(\xi,\eta)\,d\xi\,d\eta\,d\zeta \quad 1 \leq i,j \leq 4 \tag{4.72}$$

Each 3×3 matrix $\mathbf{P_{ij}}$ is constructed from a combination of the nine matrices $\mathbf{M}_{\mu,\nu}$ with its entries

$$\mathbf{P_{ij}} = \begin{bmatrix} \mathbf{M}_{\xi,\xi}(i,j) & \mathbf{M}_{\xi,\eta}(i,j) & \mathbf{M}_{\xi,\zeta}(i,j) \\ \mathbf{M}_{\eta,\xi}(i,j) & \mathbf{M}_{\eta,\eta}(i,j) & \mathbf{M}_{\eta,\zeta}(i,j) \\ \mathbf{M}_{\zeta,\xi}(i,j) & \mathbf{M}_{\zeta,\eta}(i,j) & \mathbf{M}_{\zeta,\zeta}(i,j) \end{bmatrix} \quad 1 \leq i,j \leq 4 \tag{4.73}$$

The quadratic form of the kinetic energy matrix is built up from Eq. 4.66, which is equivalent to Eq. 4.71. Each matrix $\mathbf{M}_{\mu,\nu}$ in Eq. 4.72 is solved and the elemental matrices are constructed. Now it only remains to assemble each elemental matrix entry in the global $N \times N$ kinetic energy matrix for an N node mesh.

4.3.2 Elemental Potential Matrix - Higher Dimensions

We begin by examining the two dimensional version of the potential energy matrix. It is constructed from a similar form of Eq. 4.45 or

$$\int\int \Psi_1^*(x,y)V(x,y)\Psi_1(x,y)\,dxdy \tag{4.74}$$

where the potential energy $V(x,y)$ is included in the integral. Under the change of variables as in the kinetic energy matrix, $(x,y) \to (\xi,\eta)$, the wavefunction becomes $\Psi_1(x,y) \to \Psi_1(\xi,\eta)$ along with the potential energy $V(x,y) \to V(\xi,\eta)$. How the potential energy is handled will determine the elemental potential energy matrix. A simplistic, less accurate approach is to average the potential energy within each element, $V(x,y) \to \overline{V}$. This kind of treatment would transform Eq. 4.74 into

$$\overline{V}\int\int \Psi_1^*(x,y)\Psi_1(x,y)\,dx\,dy \tag{4.75}$$

where $\overline{V}$ has been brought outside the integral. The above integral is exactly the same as the overlap integral (see Eq. 4.83) and will take on a very similar elemental matrix form. A more sophisticated approach is to assume a linear combination of the potential energy using the same basis functions defining the elemental wavefunction. If V_1, V_2, and V_3 are the values of the potential at each node in the triangular element, then we have the following linear approximation

$$V(x,y) \to V_1\phi_1(\xi,\eta) + V_2\phi_2(\xi,\eta) + V_3\phi_3(\xi,\eta) \tag{4.76}$$

$$\int_0^1\int_0^{1-\eta} (\psi_1\phi_1(\xi,\eta) + \psi_2\phi_2(\xi,\eta) + \psi_3\phi_3(\xi,\eta))^2 \cdot (V_1\phi_1(\xi,\eta) + V_2\phi_2(\xi,\eta) + V_3\phi_3(\xi,\eta))\,|J|d\xi\,d\eta \tag{4.77}$$

where the limits of integration are over the standard triangle and the Jacobian determinant $|J|$ is constant due to the linear change of variables. The integral in Eq. 4.77 represents a trilinear product of the basis functions and a bilinear product of the three unknowns ψ_i. Integrals are carried out and the quadratic form of the elemental wavefunction represented in the potential matrix is:

$$|J| \times \frac{1}{120}[\psi_1^* \quad \psi_2^* \quad \psi_3^*] \begin{bmatrix} 6V_1+2V_2+2V_3 & 2V_1+2V_2+1V_3 & 2V_1+1V_2+2V_3 \\ 2V_1+2V_2+1V_3 & 2V_1+6V_2+2V_3 & 1V_1+2V_2+2V_3 \\ 2V_1+1V_2+2V_3 & 1V_1+2V_2+2V_3 & 2V_1+2V_2+6V_3 \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \end{bmatrix} \tag{4.78}$$

Each matrix entry contains V_1, V_2, and V_3 as a weighted average. Similar to the kinetic energy matrix, the potential energy matrix is also symmetric. It only remains to assemble each elemental matrix entry in the global $N \times N$ potential energy matrix for an N node mesh. If the potential energy values $V_1 = V_2 = V_3$ in a element, the matrix is proportional to the overlap matrix or Eq. 4.75. This is an expected and necessary result. Of course, one always has the option to compute the potential energy matrix by carrying out the integral $\Psi_1^*(\xi,\eta)V(\xi,\eta)\Psi_1(\xi,\eta)$ through whatever means necessary and consider no approximation to the potential energy. However, a linear approximation to the wavefunction is already at play so it is suggested that either one of the two methods presented above is used to represent the potential.

In 3D, the potential energy matrix is built up exactly the same way as in 2D. The three dimen-

sional version of the potential energy matrix is the following

$$\int\int\int \Psi_1^*(x,y,z)V(x,y,z)\Psi_1(x,y,z)\,dx\,dy\,dz \tag{4.79}$$

where the potential energy $V(x,y,z)$ is included in the integral. Under the change of variables as in the kinetic energy matrix, $(x,y,z) \to (\xi,\eta,\zeta)$, the wavefunction becomes $\Psi_1(x,y,z) \to \Psi_1(\xi,\eta,\zeta)$ along with the potential energy $V(x,y,z) \to V(\xi,\eta,\zeta)$. Again as we saw before, how we treat the potential energy will determine the elemental potential energy matrix. The simplistic, less accurate approach is to average the potential energy within each element, $V(x,y,z) \to \overline{V}$. This kind of treatment would transform Eq. 4.79 into the overlap integral (see Eq. 4.85) and $\overline{V}$ is brought outside the integral. The second, more sophisticated approach is to assume a linear combination of the potential energy using the same basis functions defining the elemental wavefunction. If V_1, V_2, V_3, and V_4 are the values of the potential at each node in the tetrahedral element, then we have the following linear approximation

$$V(x,y,z) \to V_1\phi_1(\xi,\eta,\zeta) + V_2\phi_2(\xi,\eta,\zeta) + V_3\phi_3(\xi,\eta,\zeta) + V_4\phi_4(\xi,\eta,\zeta) \tag{4.80}$$

$$\int_0^1\int_0^{1-\eta}\int_0^{1-\eta-\zeta} (\psi_1\phi_1(\xi,\eta,\zeta) + \psi_2\phi_2(\xi,\eta,\zeta) + \psi_3\phi_3(\xi,\eta,\zeta) + \psi_4\phi_4(\xi,\eta,\zeta))^2 \cdot (V_1\phi_1(\xi,\eta,\zeta) + V_2\phi_2(\xi,\eta,\zeta) + V_3\phi_3(\xi,\eta,\zeta) + V_4\phi_4(\xi,\eta,\zeta))\,|J|d\xi\,d\eta\,d\zeta \tag{4.81}$$

where the limits of integration are over the standard tetrahedron and the Jacobian determinant $|J|$ is constant due to the linear change of variables. The integral in Eq. 4.81 represents a trilinear product of the basis functions and a bilinear product of the four unknowns ψ_i. Integrals are carried out and the quadratic form of the elemental wavefunction represented in the potential matrix is:

$$|J| \times \frac{1}{720}[\psi_1^* \quad \psi_2^* \quad \psi_3^* \quad \psi_4^*] \begin{bmatrix} (6,2,2,2) & (2,2,1,1) & (2,1,2,1) & (2,1,1,2) \\ (2,2,1,1) & (2,6,2,2) & (1,2,2,1) & (1,1,2,2) \\ (2,1,2,1) & (1,2,2,1) & (2,2,6,2) & (1,1,2,2) \\ (2,1,1,2) & (1,2,1,2) & (1,1,2,2) & (2,2,2,6) \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{bmatrix} \tag{4.82}$$

where the matrix has been simplified with the following notation: each element entry is represented by the inner product of the integer coefficients and the column vector (V_1,V_2,V_3,V_4). For example, element entry $(1,1)$ is $(6,2,2,2) \times (V_1,V_2,V_3,V_4)^t = 6\cdot V_1 + 2\cdot V_2 + 2\cdot V_3 + 2\cdot V_4$. It only remains to assemble each elemental matrix entry in the global $N \times N$ potential energy matrix for an N node mesh. If the potential energy values $V_1 = V_2 = V_3 = V_4$ in a element, the matrix is proportional to the overlap matrix. This is an expected and necessary result. Finally there is a third option, compute the potential energy matrix by carrying out the integral $\Psi_1^*(\xi,\eta,\zeta)V(\xi,\eta,\zeta)\Psi_1(\xi,\eta,\zeta)$ through whatever means necessary and consider no approximation to the potential energy. However, a linear approximation to the wavefunction is already at play so it is suggested that either one of the two methods presented above is used to represent the potential.

4.3.3 Elemental Overlap Matrix - Higher Dimensions

We begin by examining the two dimensional version of the overlap matrix. It is constructed from a similar form of Eq. 4.49 or

$$\int\int \Psi_1^*(x,y)\Psi_1(x,y)\,dx\,dy \to \int_0^1\int_0^{1-\eta} (\psi_1\phi_1(\xi,\eta) + \psi_2\phi_2(\xi,\eta) + \psi_3\phi_3(\xi,\eta))^2\,|J|\,d\xi\,d\eta \tag{4.83}$$

where again the limits of integration are over the standard triangle and we have exploited the linear change in variables with the Jacobian determinant, $|J|$, which is constant. The integral on the right hand side represents a bilinear product of the basis functions and a bilinear product of the unknowns ψ_i. The integral is carried out and the quadratic form of the elemental wavefunction represented in

the elemental overlap matrix is

$$|J| \times \frac{1}{24}[\psi_1^* \quad \psi_2^* \quad \psi_3^*] \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \end{bmatrix} \tag{4.84}$$

Similar to the kinetic and potential energy matrix, the overlap energy matrix is also symmetric. It only remains to assemble each elemental matrix entry in the global $N \times N$ overlap energy matrix for an N node mesh.

In 3D, the overlap matrix is built up exactly the same way as in 2D. The three dimensional version of the overlap matrix is the following

$$\int\int\int \Psi_1^*(x,y,z)\Psi_1(x,y,z)\,dx\,dy\,dz \rightarrow$$
$$\int_0^1 \int_0^{1-\eta} \int_0^{1-\eta-\zeta} (\psi_1\phi_1(\xi,\eta,\zeta) + \psi_2\phi_2(\xi,\eta,\zeta) + \psi_3\phi_3(\xi,\eta,\zeta) + \psi_4\phi_4(\xi,\eta,\zeta))^2\,|J|\,d\xi\,d\eta\,d\zeta \tag{4.85}$$

where again the limits of integration are over the standard tetrahedron and we have exploited the linear change in variables with the Jacobian determinant, $|J|$ which is constant. The integral on the right hand side represents a bilinear product of the basis functions and a bilinear product of the two unknowns ψ_i. The integral is carried out and the quadratic form of the elemental wavefunction represented in the elemental overlap matrix is

$$|J| \times \frac{1}{120}[\psi_1^* \quad \psi_2^* \quad \psi_3^* \quad \psi_4^*] \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{bmatrix} \tag{4.86}$$

Similar to the kinetic and potential energy matrix, the overlap energy matrix is also symmetric. It only remains to assemble each elemental matrix entry in the global $N \times N$ overlap energy matrix for an N node mesh.

4.4 Degrees of Freedom

In the previous sections, we developed FEM in the context of Schödinger's equation using linear basis functions in one, two, and three dimensions enforcing C_0 continuity. In this section, we proceed with another level of sophistication to include the possibility of requiring the wavefunction and its derivative (or inverse mass derivative) to be continuous at each node. This is C_1 continuous and requires extra degrees of freedom per node. We first define the basis functions in one, two, and three dimensions, then set up the elemental matrices required to solve Schödinger's equation in the FEM approximation within the C_1 context, and finally discuss enforcing C_1 continuity for a discontinuous effective mass as in hetrostructures.

In one dimension, we begin with the same premise as before; that is the unknown function, $\psi(x)$, is solved for using a linear combination of elemental wavefunctions and that each elemental wavefunction consists of a linear combination of basis functions. However there is an additional condition placed on the basis functions, they are chosen such that the unknown function has the value ψ_i or $\psi_i^{'}$ at node i, i.e. $\psi(x_i) = \psi_i$ or $\nabla_x\psi(x)|_i = \psi_i^{'}$. Here, $\psi_i^{'}$ is the derivative of the wavefunction amplitude at node i. This extra condition places additional constraints on the basis functions. Let's consider the benchmark line element consisting of two nodes located at $\xi_1 = -1$ and $\xi_2 = 1$ within the interval $[-1, 1]$ as before. There is an additional degree of freedom per node, totaling to four degrees of freedom per element. There is one basis function per degree of freedom, requiring four basis functions satisfying the following requirements

$$\begin{aligned} \phi_{i,0}(\xi_j) = \delta_{ij}, \quad \phi_{i,1}(\xi_j) = 0 \\ \nabla_\xi\phi_{i,0}(\xi_j) = 0, \quad \nabla_\xi\phi_{i,1}(\xi_j) = \delta_{ij} \end{aligned} \tag{4.87}$$

in order for $\psi(x_i) = \psi_i$ or $\nabla_x \psi(x)|_i = \psi_i^{'}$. The necessary interpolation polynomial needed to define each of the constraints is cubic or

$$\phi_{i,j}(\xi) = a_{i,j} + b_{i,j}\xi + c_{i,j}\xi^2 + d_{i,j}\xi^3 \quad i = 1,2\,;\, j = 0,1 \tag{4.88}$$

where a's, b's, c's, and d's are coefficients that are determined by the simultaneous equations set up by Eq. 4.87.

$$\begin{bmatrix} 1 & -1 & 1 & -1 \\ 0 & 1 & -2 & 3 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} a_{1,0} & a_{1,1} & a_{2,0} & a_{2,1} \\ b_{1,0} & b_{1,1} & b_{2,0} & b_{2,1} \\ c_{1,0} & c_{1,1} & c_{2,0} & c_{2,1} \\ d_{1,0} & d_{1,1} & d_{2,0} & d_{2,1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.89}$$

The row entries of the matrix on the left refer to the values at nodal point ξ_i evaluated for $\phi_{i,0}(\xi)$ or $\nabla_\xi \phi_{i,1}(\xi)$, the matrix on the right contains the unknown coefficients, and the matrix on the right hand side of Eq. 4.89 is just the identity matrix, which enforce the condition of Eg. 4.87. A matrix inversion determines the coefficients in each basis function.

$$\begin{aligned} \phi_{1,0}(\xi) &= \frac{2 - 3\,\xi + \xi^3}{4}, \quad \phi_{1,1}(\xi) = \frac{1 - \xi - \xi^2 + \xi^3}{4} \\ \phi_{2,0}(\xi) &= \frac{2 + 3\,\xi - \xi^3}{4}, \quad \phi_{2,1}(\xi) = \frac{-1 - \xi + \xi^2 + \xi^3}{4} \end{aligned} \tag{4.90}$$

Each elemental wavefunction consists of a linear combination of these four basis functions. Since our basis functions are in terms of local coordinates, there is some extra work required to achieve the condition $\nabla_x \psi(x)|_i = \psi_i^{'}$. To see this, let us assume that the elemental wavefunction, $\Psi_\alpha(x)$ (and its derivative $\nabla_x \Psi_\alpha(x)$) is in terms of the basis functions and amplitudes ψ_i and $\psi_i^{'}$.

$$\Psi_\alpha(x) \rightarrow \sum_{i=1}^{2} \psi_i \phi_{i,0}(\xi) + \psi_i^{'} \phi_{i,1}(\xi) \tag{4.91}$$

$$\nabla_x \Psi_\alpha(x) \rightarrow \sum_{i=1}^{2} \psi_i \frac{d\phi_{i,0}(\xi)}{d\xi}\frac{d\xi}{dx} + \psi_i^{'} \frac{d\phi_{i,1}(\xi)}{d\xi}\frac{d\xi}{dx} \tag{4.92}$$

At each of the nodes, the wavefunction collapses to the wavefunction amplitude ψ_i as required. However, the derivative of the wavefunction collapses to the product of the derivative of the wavefunction amplitude $\psi_i^{'}$ and an additional factor $\frac{d\xi}{dx}$. We have to unwind the dependence of the local coordinates. In one dimension, it is easy to just multiply a factor $\frac{dx}{d\xi}$ to $\psi_i^{'}$ in the wavefunction. The additional factor can be computed using the linear change of variables from Eq. 4.40. We consider the first element $\alpha = 1$ in a mesh that contains nodes $n = 1, 2$, so the wavefunction is

$$\Psi_1(x) \rightarrow [\phi_{1,0} \quad \phi_{2,0}] \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} + [\phi_{1,1} \quad \phi_{2,1}] \frac{dx}{d\xi} \begin{bmatrix} \psi_1^{'} \\ \psi_2^{'} \end{bmatrix} \tag{4.93}$$

Now that the wavefunction has been represented in terms of unknown quantities ψ_i and $\psi_i^{'}$. We can proceed by following in the footsteps of the previous prescription. Each elemental matrix is built up and assembled in the respective $2N \times 2N$ global matrix containing N nodes.

In two and three dimensions, we follow a flow of ideas similar to those presented above for one dimension. Basis functions are defined for the standard triangle in local coordinates (ξ, η) and standard tetrahedron in local coordinates (ξ, η, ζ) for two and three dimensions respectively. All elements are linearly mapped to the basis functions through the same linear transformations defined for C_0 continuity (see Eq. 4.55 and 4.59).

For each triangular element, we introduce three unknowns per node identified as

$$\psi_{i,0} = \psi|_i \quad \psi_{i,1} = \frac{\partial \psi}{\partial x}|_i \quad \psi_{i,2} = \frac{\partial \psi}{\partial y}|_i \tag{4.94}$$

and the basis functions are chosen such that the wavefunction has the values $\psi_{i,0}$, $\psi_{i,1}$, or $\psi_{i,2}$ at node i. These extra conditions place additional constraints on the basis functions. Three degrees of freedom per node and three nodes per triangular element totals nine degrees of freedom per element. Each degree of freedom requires a basis function and so there will be nine basis functions per element. The basis functions are constructed in the local coordinate system and must satisfy the following constraints

$$\begin{aligned} \phi_{i,0}(\xi_j,\eta_j) = \delta_{ij}, \quad \phi_{i,1}(\xi_j,\eta_j) = 0, \quad \phi_{i,2}(\xi_j,\eta_j) = 0 \\ \nabla_\xi \phi_{i,0}(\xi_j,\eta_j) = 0, \quad \nabla_\xi \phi_{i,1}(\xi_j,\eta_j) = \delta_{ij}, \quad \nabla_\xi \phi_{i,2}(\xi_j,\eta_j) = 0 \\ \nabla_\eta \phi_{i,0}(\xi_j,\eta_j) = 0, \quad \nabla_\eta \phi_{i,1}(\xi_j,\eta_j) = 0, \quad \nabla_\eta \phi_{i,2}(\xi_j,\eta_j) = \delta_{ij} \end{aligned} \tag{4.95}$$

The necessary interpolation polynomial needed to define each constraint is cubic

$$\begin{aligned} \phi_{i,j}(\xi,\eta) = a_{i,j} + b_{i,j}\,\xi + c_{i,j}\,\eta + d_{i,j}\,\xi^2 + e_{i,j}\,\xi\eta + f_{i,j}\,\eta^2 + g_{i,j}\,\xi^3 + \\ h_{i,j}\,\xi^2\eta + k_{i,j}\,\xi\eta^2 + l_{i,j}\,\eta^3 \;\; ; i = 1,2,3\,;\; j = 0,1,2 \end{aligned} \tag{4.96}$$

where the coefficients are determined by the simultaneous equations set up by Eq. 4.95. This sets up a 9×10 matrix, $\mathbf{A}$, containing values at nodal point (ξ_i, η_i) evaluated for $\phi_{i,0}(\xi_j,\eta_j)$, $\nabla_\xi \phi_{i,1}(\xi_j,\eta_j)$, and $\nabla_\eta \phi_{i,2}(\xi_j,\eta_j)$ multiplied by a 10×9 matrix, $\mathbf{B}$, containing the to be determined coefficients equaling a 9×9 identity matrix, $\mathbf{I}$. A matrix inversion of $\mathbf{A}$ will determine the coefficients. However, a small problem exists. It is difficult to perform a matrix inversion on a 9×10 matrix. To get around this, we compute the *kernel* of the matrix $\mathbf{A}$. This is a vector that is orthogonal to all the nine rows in this matrix. The kernel consists of one vector with 10 components. It is unique up to sign and scale. This row vector is appended to the bottom of matrix $\mathbf{A}$ to make a 10×10 nonsingular matrix. Now matrix $\mathbf{A}$ can be inverted and the basis functions are determined.

$$\begin{aligned} \phi_{1,0} &= 1 - \phi_{2,0} - \phi_{3,0}, \\ \phi_{1,1} &= \xi - \phi_{2,0} - \phi_{2,1} - \phi_{3,1}, \\ \phi_{1,2} &= \eta - \phi_{2,2} - \phi_{3,0} - \phi_{3,2}, \\ \phi_{2,0} &= 3\xi^2 - 2\xi^3, \\ \phi_{2,1} &= -\xi^2 + \xi^3, \\ \phi_{2,2} &= \frac{1}{3}\left(\xi\eta + 2\xi^2\eta - \xi\eta^2\right), \\ \phi_{3,0} &= 3\eta^2 - 2\eta^3 \\ \phi_{3,1} &= \frac{1}{3}\left(\xi\eta + 2\xi\eta^2 - \xi^2\eta\right) \\ \phi_{3,2} &= -\eta^2 + \eta^3 \\ \phi_{10} &= \frac{1}{3}\left(-\xi\eta + \xi^2\eta + \xi\eta^2\right) \end{aligned} \tag{4.97}$$

With this procedure, an additional basis function ϕ_{10} is created whose wavefunction values and first derivatives vanish at all three vertices. Due to these properties, this extra basis function could be used to reduce the complexity of the other nine basis functions by an addition or subtraction.

Each elemental wavefunction consists of a linear combination of these nine basis functions. Since our basis functions are in terms of local coordinates, there is some extra work required to achieve the condition $\psi_{i,1}$, or $\psi_{i,2}$ at node i. In order to unwind the local coordinate dependence, we use

the Jacobian (**J**) to relate

$$\begin{bmatrix} \phi_{i,1}(\xi,\eta) \\ \phi_{i,2}(\xi,\eta) \end{bmatrix} = \mathbf{J} \begin{bmatrix} \psi_{j,1}(x,y) \\ \psi_{j,2}(x,y) \end{bmatrix} \tag{4.98}$$

where i, j index the vertices of a triangular element in real space and the local vertices of the benchmark triangle that it is mapped to. The Jacobian's entries can be computed using the linear change of variables from Eq. 4.55. We consider the first element $\alpha = 1$ in a mesh that contains nodes $n = 1, 2, 3$, so the elemental wavefunction is

$$\Psi_1(x,y) \to [\phi_{1,0} \quad \phi_{2,0} \quad \phi_{3,0}] \begin{bmatrix} \psi_{1,0} \\ \psi_{2,0} \\ \psi_{3,0} \end{bmatrix} + \sum_{i=1}^{3} [\phi_{i,1} \quad \phi_{i,2}] \, \mathbf{J} \begin{bmatrix} \psi_{i,1} \\ \psi_{i,2} \end{bmatrix} \tag{4.99}$$

Now the wavefunction has been represented in terms of unknown quantities $\psi_{i,0}$, $\psi_{i,1}$, and $\psi_{i,2}$. We proceed by following in the footsteps of the previous prescription. Each elemental matrix is built up and assembled in the respective $3N \times 3N$ global matrix containing N nodes.

For each tetrahedral element, we introduce four unknowns per node i identified as

$$\psi_{i,0} = \psi|_i \quad \psi_{i,1} = \frac{\partial \psi}{\partial x}|_i \quad \psi_{i,2} = \frac{\partial \psi}{\partial y}|_i \quad \psi_{i,3} = \frac{\partial \psi}{\partial z}|_i \tag{4.100}$$

and the basis functions are chosen such that the wavefunction has the values $\psi_{i,0}$, $\psi_{i,1}$, $\psi_{i,2}$, or $\psi_{i,3}$ at node i. These extra conditions place additional constraints on the basis functions. Four degrees of freedom per node and four nodes per tetrahedral element totals 16 degrees of freedom per element. Each degree of freedom requires a basis function and so there will be 16 basis functions per element. The basis functions are constructed in the local coordinate system and must satisfy the following constraints

$$\begin{gathered}
\phi_{i,0}(\xi_j,\eta_j,\zeta_j) = \delta_{ij}, \quad \phi_{i,1}(\xi_j,\eta_j,\zeta_j) = 0, \quad \phi_{i,2}(\xi_j,\eta_j,\zeta_j) = 0, \quad \phi_{i,3}(\xi_j,\eta_j,\zeta_j) = 0 \\
\nabla_\xi \phi_{i,0}(\xi_j,\eta_j,\zeta_j) = 0, \quad \nabla_\xi \phi_{i,1}(\xi_j,\eta_j,\zeta_j) = \delta_{ij}, \quad \nabla_\xi \phi_{i,2}(\xi_j,\eta_j,\zeta_j) = 0, \quad \nabla_\xi \phi_{i,3}(\xi_j,\eta_j,\zeta_j) = 0 \\
\nabla_\eta \phi_{i,0}(\xi_j,\eta_j,\zeta_j) = 0, \quad \nabla_\eta \phi_{i,1}(\xi_j,\eta_j,\zeta_j) = 0, \quad \nabla_\eta \phi_{i,2}(\xi_j,\eta_j,\zeta_j) = \delta_{ij}, \quad \nabla_\eta \phi_{i,3}(\xi_j,\eta_j,\zeta_j) = 0 \\
\nabla_\zeta \phi_{i,0}(\xi_j,\eta_j,\zeta_j) = 0, \quad \nabla_\zeta \phi_{i,1}(\xi_j,\eta_j,\zeta_j) = 0, \quad \nabla_\zeta \phi_{i,2}(\xi_j,\eta_j,\zeta_j) = 0, \quad \nabla_\zeta \phi_{i,3}(\xi_j,\eta_j,\zeta_j) = \delta_{ij}
\end{gathered} \tag{4.101}$$

The necessary interpolation polynomial needed to define each constraint is cubic constructed from the monomials $\xi^p \eta^q \zeta^s$, $p \geq 0, q \geq, s \geq 0, p + q + s \leq 3$ with a similar structure as the two dimensional version, Eq. 4.96. There are 20 coefficients per basis functions and are determined by the simultaneous equations set up by Eq. 4.101. This sets up a 16×20 matrix, **A**, containing values at nodal point (ξ_i, η_i, ζ_i) evaluated for $\phi_{i,0}(\xi_j, \eta_j)$, $\nabla_\xi \phi_{i,1}(\xi_j, \eta_j)$, $\nabla_\eta \phi_{i,2}(\xi_j, \eta_j)$, and $\nabla_\zeta \phi_{i,3}(\xi_j, \eta_j)$ multiplied by a 20×16 matrix, **B**, containing the to be determined coefficients equaling a 16×16 identity matrix, **I**. Before matrix **A** is inverted to determine the coefficients making up the 16 basis functions, we compute the kernal of matrix **A**. There are 4 vectors that are orthogonal to all the 16 rows. These 4 row vectors are appended to the bottom of matrix **A** to make a 20×20 nonsingular matrix. Now matrix **A** can be inverted and the basis functions are determined. Table 4.2 summarizes the 16 basis functions. It should be noted that the basis functions have been reduced in complexity by exploiting the extra functions, $\phi_{17-19} = -\frac{1}{3} x_i x_j + \frac{1}{3} x_i^2 x_j + \frac{1}{3} x_i x_j^2$ $(1 < i < j < 3)$, that were generated in the process.

Each elemental wavefunction consists of a linear combination of these 16 basis functions. Since our basis functions are in terms of local coordinates, there is some extra work required to achieve the condition $\psi_{i,1}$, $\psi_{i,2}$, or $\psi_{i,3}$ at node i. In order to unwind the local coordinate dependence, we use the Jacobian (**J**) as before. We consider the first element $\alpha = 1$ in a mesh that contains nodes

Table 4.2: The C_1 basis functions that for the right tetrahedron with vertices located at (0,0,0), (1,0,0), (0,1,0), and (0,0,1).

(0,0,0)	(1,0,0)	(0,1,0)	(0,0,1)
$\phi_{1,0} = 1 - \phi_{2,0} - \phi_{3,0} - \phi_{4,0}$	$\phi_{2,0} = 3\xi^2 - 2\xi^3$	$\phi_{3,0} = 3\eta^2 - 2\eta^3$	$\phi_{4,0} = 3\zeta^2 - 2\zeta^3$
$\phi_{1,1} = \xi - \phi_{2,0} - \phi_{2,1} - \phi_{3,1} - \phi_{4,1}$	$\phi_{2,1} = -\xi^2 + \xi^3$	$\phi_{3,1} = \xi\eta^2$	$\phi_{4,1} = \xi\zeta^2$
$\phi_{1,2} = \eta - \phi_{2,2} - \phi_{3,0} - \phi_{3,2} - \phi_{4,2}$	$\phi_{2,2} = \xi^2\eta$	$\phi_{3,2} = -\eta^2 + \eta^3$	$\phi_{4,2} = \eta\zeta^2$
$\phi_{1,3} = \zeta - \phi_{2,3} - \phi_{3,3} - \phi_{4,0} - \phi_{4,3}$	$\phi_{2,3} = \xi^2\zeta$	$\phi_{3,3} = \eta^2\zeta$	$\phi_{4,3} = -\zeta^2 + \zeta^3$

$n = 1, 2, 3, 4$, so the elemental wavefunction is

$$\Psi_1(x,y) \to [\phi_{1,0} \quad \phi_{2,0} \quad \phi_{3,0} \quad \phi_{4,0}] \begin{bmatrix} \psi_{1,0} \\ \psi_{2,0} \\ \psi_{3,0} \\ \psi_{4,0} \end{bmatrix} + \sum_{i=1}^{4} [\phi_{i,1} \quad \phi_{i,2} \quad \phi_{i,3}]\, \mathbf{J} \begin{bmatrix} \psi_{i,1} \\ \psi_{i,2} \\ \psi_{i,3} \end{bmatrix} \tag{4.102}$$

Now the wavefunction has been represented in terms of unknown quantities $\psi_{i,0}$, $\psi_{i,1}$, $\psi_{i,2}$, and $\psi_{i,3}$. We proceed by following in the footsteps of the previous prescription. Each elemental matrix is built up and assembled in the respective $4N \times 4N$ global matrix containing N nodes.

As we have seen, the complexity of the elemental wavefunctions increases due to enforcing C_1 continuity at each node. Additionally, the elemental matrices will be larger due to the increased number of basis functions than its C_0 counterpart. However, the prescription for constructing each of the elemental matrices outlined above for one, two, and three dimensions is generally the same. There might be one additional step that is needed to remove the unknown derivative amplitude's dependence on the local coordinates if the Jacobian in Eq. 4.99 and Eq. 4.102 is not directly included in the elemental wavefunction. An example of this additional step is found in the MATLAB program described in the next chapter.

In C_1 continuity, we are requiring that the wavefunction and its derivative be continuous at each node. As we saw in the derivation of Eq. 4.4, for a spatially varying mass, the wavefunction and its inverse mass derivative should be continuous. So for nodes falling on the interface boundary for two different effective masses, we should employ the continuity of the wavefunction and its inverse mass derivative. This can be easily accomplished in FEM by rescaling the derivatives of the nodes falling on the interface with the proper ratio of masses in the elemental matrix before it is overlaid in the global matrix. This entails multiplying the corresponding row and column by the appropriate ratio. To be explicit, lets consider the one dimensional quantum well example from Sec. 4.1 except that there are only 3 elements and 4 nodes, see Fig. 4.3. Elements 1 and 3 represent barrier material with an effective mass of m_b and element 2 represents the quantum well material with an effective mass of m_w. Nodes 2 and 3 fall on the interface between the materials. We use the form of the wavefunction from Eq. 4.93 with a slight variation on the nomenclature such that Ψ^α refers to the elemental wavefunction and ψ_i^α refer to the unknown coefficients at each node. In addition, the factor $\frac{d\xi}{dx}$ is absorbed in $\psi_i^{\alpha'}$ such that

$$\begin{aligned} \Psi^1(x) &\to \psi_1^1\phi_{1,0} + \psi_2^1\phi_{2,0} + \psi_1^{1'}\phi_{1,1} + \psi_2^{1'}\phi_{2,1} \\ \Psi^2(x) &\to \psi_1^2\phi_{1,0} + \psi_2^2\phi_{2,0} + \psi_1^{2'}\phi_{1,1} + \psi_2^{2'}\phi_{2,1} \\ \Psi^3(x) &\to \psi_1^3\phi_{1,0} + \psi_2^3\phi_{2,0} + \psi_1^{3'}\phi_{1,1} + \psi_2^{3'}\phi_{2,1} \end{aligned} \tag{4.103}$$

Now we impose the continuity conditions across the interface

$$\begin{aligned} \psi_2^\alpha &= \psi_1^{(\alpha+1)} \\ \frac{1}{m_\alpha}\psi_2^{\alpha'} &= \frac{1}{m_{\alpha+1}}\psi_1^{(\alpha+1)'} \to \psi_1^{(\alpha+1)'} = \frac{m_{\alpha+1}}{m_\alpha}\psi_2^{\alpha'} \end{aligned} \tag{4.104}$$

where $\alpha = 1, 2$. These conditions transform the wavefunctions into the following

$$\begin{aligned}
\Psi^1(x) &\to \psi_1^1\phi_{1,0} + \psi_2^1\phi_{2,0} + \psi_1^{1'}\phi_{1,1} + \psi_2^{1'}\phi_{2,1} \\
\Psi^2(x) &\to \psi_2^1\phi_{1,0} + \psi_2^2\phi_{2,0} + \frac{m_w}{m_b}\psi_2^{1'}\phi_{1,1} + \psi_2^{2'}\phi_{2,1} \\
\Psi^3(x) &\to \psi_2^2\phi_{1,0} + \psi_2^3\phi_{2,0} + \frac{m_b}{m_w}\psi_2^{2'}\phi_{1,1} + \psi_2^{3'}\phi_{2,1}
\end{aligned} \tag{4.105}$$

or equivalently

$$\begin{aligned}
\Psi^1(x) &\to \psi_1^1\phi_{1,0} + \psi_2^1\phi_{2,0} + \psi_1^{1'}\phi_{1,1} + \psi_2^{1'}\phi_{2,1} \\
\Psi^2(x) &\to \psi_2^1\phi_{1,0} + \psi_1^3\phi_{2,0} + \frac{m_w}{m_b}\psi_2^{1'}\phi_{1,1} + \frac{m_w}{m_b}\psi_1^{3'}\phi_{2,1} \\
\Psi^3(x) &\to \psi_1^3\phi_{1,0} + \psi_2^3\phi_{2,0} + \psi_1^{3'}\phi_{1,1} + \psi_2^{3'}\phi_{2,1}
\end{aligned} \tag{4.106}$$

The above equations are easier to implement for computational purposes, especially in higher dimensions. Each elemental matrix is constructed from Eq. 4.106 using the outlined prescription and then overlaid in each respective global matrix. For higher dimensions, we use the following procedure: (1)for each element, determine if any nodes are located on the interface boundary AND if the element falls inside the quantum wire or quantum dot structure, (2) construct each elemental matrix as normal, (3) for those nodes found from (1), multiply the row AND column corresponding to the derivative by the appropriate ratio, and (4) finally place matrix elements in respective global matrix.

4.5 Remarks

In this chapter, we have developed FEM in the context of Schödinger's equation with the sophistication required to find bound state energy levels and wavefunctions associated with a confining potential in one, two, and three dimensions. We introduced the Schödinger's equation in its variational form and derived the method using a linear combination of elemental wavefunctions. The quadratic forms of the wavefunction were represented for the kinetic, potential, and overlap matrices in one dimension and applied to the infinite potential well to benchmark its usefulness and accuracy.

We then proceeded to derive basis functions for higher dimensions and higher degrees of freedom per node. We followed the prescription identified in Sec. 4.2 with appropriate adjustments. For each added complexity, we developed the necessary quadratic forms of the wavefunction in the form of the kinetic, potential, and overlap elemental matrices. Finally, we addressed the ability of FEM to incorporate the boundary conditions for a spatially varying mass when considering C_1 continuity.

Chapter 5: Finite Element Program

In the previous chapter, FEM was introduced as method that transforms Schrödinger's equation into matrix mechanics. Once in matrix form, solutions are obtained through various operations that normally involve matrix inversion and matrix diagonalization. As such, computer programs utilizing modern linear algebra algorithms are used to perform these operations fairly quickly. Today, there are many technical computing software packages that include these algorithms as efficient built-in functions. This allows for quick efficient programming. MATLAB is one such software package that includes a large built-in linear algebra library making it a suitable environment to write FEM code [52]. In addition to technical computing software, there are many commercial FEM software packages that advertise solving everything from heat flow to stock option pricing [53, 54, 55, 56]. However, most of these packages are developed for the engineer, are customized and difficult to modify, and expensive.

In this chapter, we begin in Sec. 5.1 with an introduction on how a typical FEM program is organized, then proceed to Secs. 5.2, 5.3, and 5.4 with a detailed description of the entire MATLAB FEM program that is used to determine energy levels and corresponding wavefunctions of quantum heterostuctures in the effective mass approximation. We then validate the FEM program with benchmarked solutions in Sec. 5.5 and conclude in Sec. 5.6. Throughout these sections, relevant source code is displayed to supplement the discussions, while the entire FEM program source code is given in Appendix B.

5.1 Stages of FEM Programming

There are a few commonalities that exist in all FEM code no matter what problem the code is aiming to solve. There is the preprocessing stage that creates the mesh for the particular problem. This stage generates the nodal coordinates, elements, and boundary information in data structures used in the next step of the program called the processing stage. The processing stage is where the various matrices of the problem are populated or assembled, i.e. kinetic, potential, and overlap matrices for Schrödinger's equation. Accordingly, the processing stage is the heart of the finite element method. Finally, there is the post-processing stage that displays the solution of the problem with possibly some sort of visualization. Fig. 5.1 shows a flowchart of the three stages required in an FEM program. The output from each stage is shown above the arrows and used as the input in the next stage.

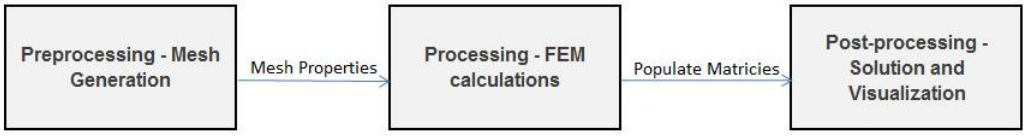


Figure 5.1: A flowchart for finite element programs.

5.2 FEM Program - Preprocessing Code

This part of the program defines the physical domain of the problem and tessellates it into elements and nodes. This is not as trivial as one might think and requires experience rather than a defined set of rules. To illustrate, consider the domain of a particle in a finite square potential well that has dimensions of $-L < x, y < L$ and we are searching for bound states. Part of the wavefunction

will exist outside the well and asymptotically decay to zero. That means that the physical domain theoretically extends to infinity. The user will have decide where the wavefunction is 'sufficiently' small to apply the asymptotic boundary conditions. This can be tricky for a few reasons:

- In general, a wavefunction will extend further outside the well as the energetic state increases; and
- The magnitude of the potential well depth effects how far wavefunctions extend outside the well (compare to infinite wells).

Further, after the physical domain is defined, the user has to decide what relative size the element's edges should be. The edges ultimately determine the number of nodes and elements in the mesh. As we saw in the Chapter 4, the accuracy of the solution improves with an increase in the number of elements but this comes at an added computational cost. The solution to this accuracy versus cost dilemma isn't always easy and ultimately depends on the specific problem.

Before we dig any deeper, it is useful to understand how the nodal and elemental information is stored for input into the processing stage. Unsurprisingly, node information is organized in an $nnodes \times ndim$ matrix. The variables $nnodes$ and $ndim$ refer to the number of nodes and dimensions of the domain respectively with each row corresponding to the node number and each column corresponding to the respective coordinate. Element information is organized in a 'connectivity' matrix which tells the processing stage how nodes in each element are connected. The size of the matrix is $nel \times ndim + 1$[1], with nel number of elements. To illustrate how these matrices are organized, consider the simple 2D mesh in Fig. 5.2, which consists of four nodes tessellated into two triangular elements. If the coordinates of the nodes are as follows; node 1 $(0, 0)$, node 2 $(1, 0)$, node 3 $(1, 1)$, and node 4

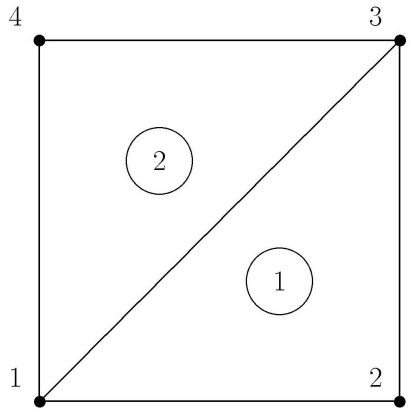


Figure 5.2: A simple mesh containing four nodes and two triangular elements.

$(0, 1)$, then node matrix is simply Eq. 5.1.

$$nodes = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \tag{5.1}$$

The connectivity matrix contains the node numbers that represent vertices of each triangular element in Fig. 5.2 and shown in Eq. 5.2. Notice the elements are all ordered in a counter - clockwise fashion. This is done so the Jacobians will all be positive.

$$Connect = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 4 \end{bmatrix} \tag{5.2}$$

[1]This will be true for triangular elements in 2D and tetrahedral elements in 3D.

The main pre-processing code is built upon a freely distributed mesh package called **Distmesh** [50] that applies a mechanical analogy between elements and a truss structure. Edges of the elements are considered bars and the nodes are considered the joints connecting the bars. External forces are applied to the boundary of the truss that produces reactionary forces acting on the joints. The algorithm solves for equilibrium, iteratively moving the joints and adjusting the topology based on a technique known as Delaunay triangulation [57]. When the movement of the joints after an iteration is smaller than a default tolerance, the algorithm is terminated. Those are essentially the steps used in creating the mesh. Although the code is a few dozen lines long, it is powerful in the sense that it is easy to understand and intuitive to build upon. The package comes with various MATLAB function files that supplement the main mesh generating functions (2D version called *distmesh2d* or ND version called *distmeshnd*) and used in the development of our mesh generating pre-processing code.

One of the key designs the authors used in developing the Distmesh software was to represent the mesh geometry by defining a signed distance function that is negative inside the represented boundary. As the name suggests, the distance function determines a signed distance from each of the nodes to the geometry boundary. Our mesh generating code utilizes this distance function formulation.

Distmesh also comes with some predefined special distance functions in 2D that can be combined to create more complex mesh geometries. This is especially convenient when defining geometries that could be attributed to quantum wires. However, defining distance functions in 3D is more difficult and Distmesh does not offer the same amenities as it does in 2D. Since we are most interested in quantum dot geometries, we spent a considerable amount of effort developing efficient polyhedron distance functions. This work resulted in a template that is used to generate polyhedron meshes. We will explain how to use this template and demonstrate its use through worked examples.

The final mesh must represent both the quantum and barrier material due to exponentially decaying wavefunction penetration into the barrier region. As such, the mesh must be cutoff at distances into the barrier region where the wavefunction is 'sufficiently' enough to satisfy the boundary conditions 'at infinities'. It is convenient for the cutoff distances to fall on either the edges of a square or the surfaces of a cube. This is not a requirement but a recommendation due to the simplicity of representing a square or cube in the distance function formulation. It should be recognized that this recommendation might not generate the most efficient mesh. In the context of FEM, an efficient mesh contains fewer elements than another mesh containing more elements while producing similar results with the same degree of accuracy. Ultimately, the decision on how to represent the barrier geometry should be based on the specific problem at hand while taking into consideration 'simplicity' and 'mesh efficiency'.

The material interface in the final mesh must separate the quantum material from the barrier material such that every element is located in one region or the other. This requires every node in an element to be located either in the quantum region, in the barrier region, or on the boundary between them. If this were not the case, the geometry of the quantum material would be distorted. Further, the edges of certain elements near the material interface must be constrained to this internal boundary. This is called a constrained mesh, see Fig. 5.3, and can be difficult to create due to the underlying algorithm that tessellates the set of points or nodes into simplices (either triangles or tetrahedrons). As mentioned, the built-in algorithm is based on Delaunay tessellation which triangulates a set of points $\mathbf{P}$ in a plane such that no point in $\mathbf{P}$ is inside the circumsphere of any triangle. This method was invented in 1934 by Boris Delaunay [57]. By considering circumhyperspheres[2], the notion of Delaunay tessellation is extended to higher dimensions quite easily. Since the set of nodes in the final mesh represents both the quantum and barrier material, calling the Delaunay algorithm might generate elements near the interface containing nodes in both regions, effectively distorting the region. This can be remedied a number of ways. One common way is to create "non-Delaunay" elements after the algorithm is called by flipping certain edges or surfaces near the interface. Another way is to create two or more meshes separately, each of which has the desired interface geometry, combine and strategically eliminate certain nodes before the algorithm is called. This latter idea is

[2] A cicumhypersphere is a hypersphere that contains the polyhedron and touches each of the polyhedron's vertices.

used in our code to create the constrained mesh.

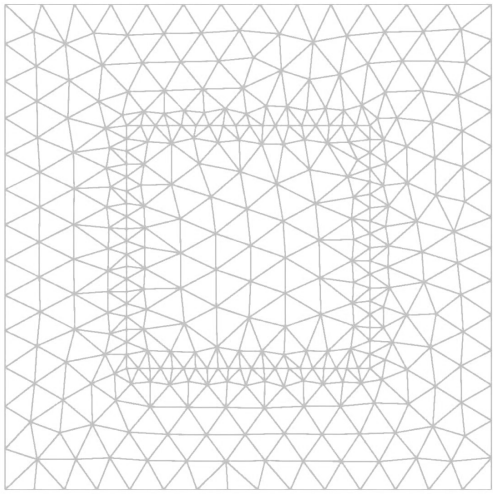


Figure 5.3: A constrained mesh with refinement around the interface defining the cross section geometry of a square quantum wire.

It is also desirable to increase the number of elements near the interface, as seen in Fig. 5.3. An increased ratio of elements near the interface to the total number of elements in the mesh will increase the number of evaluations made near the two materials. This improves accuracy since unique solutions to differential equations are defined by their boundary conditions. An additional benefit occurs when the interface between the two materials is curved, i.e. spherical, since linear edges are being used to fit the curved space.

5.2.1 Implementation

The source code for the 2D mesh preprocessor is contained in Fig. 5.4. Comments within the code are preceded by % and each line of the code is explained in detail below.

The first line of the code specifies the calling sequence of the syntax for the function QWire_mesh:

```
function [p,t,vert] = QWire_Mesh( QW_fd,B_fd,fh,h0,QW_bbox,B_bbox )
```

This 2D mesh function produces the following output:

- The node positions **p**. This $N \times 2$ matrix contains the x and y positions for each of the N nodes.
- The triangle vertices **t**. This $NT \times 3$ matrix contains the 3 vertices, each referring to the respective node number, for each of the NT triangles in the mesh.
- The nodes on the interface **vert**. This vector contains the nodes that fall on the interface.

The input arguments are the following:

- The quantum material geometry is given as a distance function **QW_fd**.
- The barrier material geometry is given as a distance function **B_fd**.
- The relative edge length is given as a function **fh**.
- The initial edge length is given by the scalar **h0**.
- The bounding box of the quantum material is given as a 2×2 matrix **QW_bbox** $= [x_{min}, y_{min}; x_{max}, y_{max}]$.

```
function [p,t,vert] = QWire_Mesh( QW_fd,B_fd,fh,h0,QW_bbox,B_bbox )

% Create a mesh for the quantum wire material using distmesh2D
[QW_coord,QW_nodes]=distmesh2d(QW_fd,fh, h0,QW_bbox,[]);
[QW_coord,QW_nodes]=fixmesh(QW_coord,QW_nodes);
QW_surface=unique(boundedges(QW_coord,QW_nodes));
fixed=[QW_coord(QW_surface(:),1),QW_coord(QW_surface(:),2)];
% Create a mesh for the barrier material using distmesh 2D
%
[B_coord,B_nodes]=distmesh2d(B_fd,fh,h0,B_bbox,fixed);
[B_coord,B_nodes]=fixmesh(B_coord,B_nodes);
B_surface=unique(boundedges(B_coord,B_nodes));
% Convert coordinates to desired significant digits
%
B_coord=round(B_coord*1000)/1000;
QW_coord=round(QW_coord*1000)/1000;
fixed=round(fixed*1000)/1000;
vert = fixed;
%
% 1. Get coordinates of all the nodes on edges of the barrier material
edges=[B_coord(B_surface(:),1),B_coord(B_surface(:),2)];
%
% 2. Start to systematically eliminate nodes associated with outer edges
for i=1:2
    for j=1:2
        [r c]=find(edges == B_bbox(i,j));
        edges(r,:)=[];
    end
end
%
% 3. Start to systematically eliminate nodes associated with nodes on
% the QW edges.
[index]=ismember(edges,fixed,'rows');
edges(index,:)=[];
%
% 4. Eliminate "orphan" nodes
[index]=ismember(B_coord,edges,'rows');
B_coord(index,:)=[];
%
% 5. Complete the final mesh
p=unique([B_coord;QW_coord],'rows');
t=delaunayn(p);
%
% 6. View the final mesh
trimesh(t,p(:,1),p(:,2),zeros(size(p,1),1))
view(2),axis equal,axis off,drawnow
end
```

Figure 5.4: The source code for generating a two dimensional quantum wire mesh called QWire_mesh. Comments are preceded by %.

- The bounding box of the barrier material is given as a 2×2 matrix **QW_bbox** $= [x_{min}, y_{min}; x_{max}, y_{max}]$.

In the beginning of the code, before the numbered steps begin, the quantum mesh is created by calling *distmesh2d* using the distance function **QW_fd**. The mesh is 'cleaned up' by calling the function *fixmesh* which eliminates unused or duplicate nodes and orients the elements correctly. Following the clean up, the nodes on the edges of the mesh are stored in the array **QW_surface** by calling the function *boundedges* and their coordinates are stored in the matrix **fixed**. Ultimately, the nodes stored in the **QW_surface** array will be the only nodes located on the barrier/quantum material interface.

Following the quantum mesh, the barrier mesh is created by calling *distmesh2d* using the distance function **B_fd** and supplying the matrix **fixed** as fixed node positions. Using the same approach, the barrier mesh is cleaned up and nodes on the edges are stored in the array **B_surface**. Next, the coordinates of the nodes corresponding the quantum mesh (**QW_coord**), barrier mesh (**B_coord**), and nodes on the interface (**fixed**) are converted to desired significant digits.

Now we describe the numbered steps 1 to 6 as indicated in Fig. 5.4.

1. The first step stores the coordinates of all the nodes that fall on the edges of the barrier mesh:

```
edges=[B_coord(B_surface(:),1),B_coord(B_surface(:),2)];
```

The first column of the matrix edges stores the x coordinate and the second column stores the y coordinate.

2. Next, we start to systematically eliminate the nodes from the matrix **edges** that fall on the outer edges (the barrier mesh will have the quantum material geometry 'cut out' of its center, see 5.2.2):

```
for i=1:2
for j=1:2
[r c]=find(edges == B_bbox(i,j));
edges(r,:)=[];
end
end
```

The for loops used in conjunction with the function *find* are used to search for and store row indices from matrix **edges** that fall on the bounding box. Those rows are then deleted from **edges**.

3. The next piece of the code eliminates the nodes from the matrix **edges** that were fixed in the barrier mesh. These nodes are equivalent to the nodes located on the edges of the quantum mesh:

```
[index]=ismember(edges,fixed,'rows');
edges(index,:)=[];
```

The function *ismember* with the option 'rows' returns the vector **index** containing 1 where the rows of **edges** are also rows of **fixed** and 0 otherwise. The rows in **edges** that correspond to 1 in **index** are deleted.

4. Now, the coordinates stored in matrix **edges** only correspond those nodes on the boundary of the barrier mesh that fall on the interface but are not commonly shared by the quantum mesh. For the purposes of this work, they are called 'orphan' nodes and are eliminated from the barrier mesh:

```
[index]=ismember(B_coord,edges,'rows');
B_coord(index,:)=[];
```

Again the function *ismember* with the option 'rows' is used to return the vector **index** containing 1 where the rows of **B_coord** are also rows **edges** and 0 otherwise. The rows in **B_coord** that correspond to 1 in **index** are deleted.

5. We are now in a position to combine the quantum and barrier meshes into the final mesh:

```
p=unique([B_coord;QW_coord],'rows');
t=delaunayn(p);
```

First, the nodes of the final matrix are stored in matrix **p** by combining **B_coord** and **QW_coord** and eliminating similar nodes on the interface by calling the function *unique* with the option 'rows'. The final matrix **p** represents the matrix referred as **Nodes** in Eq. 5.1. Next, the connectivity matrix **t** is constructed using the coordinates **p** in conjunction with the function *delaunayn* which calls the Delaunay algorithm. The final matrix **t** represents the matrix referred as **Connect** in Eq. 5.2. Requiring the barrier mesh to contain those nodes on the quantum mesh boundary and eliminating the orphan nodes from the barrier matrix guarantees the interface nodes will be those that originally fell on the quantum mesh boundary. This method creates a constrained mesh after calling the Delaunay algorithm because it reduces the likelihood there will be a circumhypersphere containing no points in the set **p** spanning the interface. On occasion this method might fail when the number of elements near the interface are not increased relative to the rest of the mesh (see relative edge length function example in Sec. 5.2.2. This is a simple, yet effective method for creating a constrained mesh in both 2D and 3D.

6. The last part of the code is to display the final mesh:

```
trimesh(t,p(:,1),p(:,2),zeros(size(p,1),1))
view(2),axis equal,axis off,drawnow
end
```

The source code for the 3D mesh preprocessor is contained in Fig. 5.5. As expected, the source code is similar to the 2D mesh preprocessor with each line of code performing those same functions as expressed above. The numbered steps are synchronized to those steps in the 2D mesh preprocessor so that a detailed explanation of each line is not necessary.

The preprocessing code QWire_mesh and Qdot_mesh is all that is needed to mesh quantum wires and dots consisting of complex geometries when the proper distance functions are supplied. As mentioned, the Distmesh package comes with some special distance functions that can be used to generate a more complicated mesh. They include rectangular, circular, and polygonal 2D distance functions and some other special functions that combine geometries. As will be seen in Sec. 5.2.2, these already defined distance functions are normally sufficient for supplying geometries to Qwire_mesh.

The distance functions needed to supply Qdot_mesh are inherently more difficult to construct. As an example, consider defining the distance function for a polyhedron. The code would have to compute distances of points to line segments and enclosed faces in 3D while trying to avoid inefficient computer loops. Since the Distmesh package only comes with the distance function for a sphere, we have written code for the rectangular prism, five sided pyramid and a template for any polyhedron.

1. The distance function for the rectangular prism:

```
function d_signed=drectangle_3D(p,pv)
%drectangle_3D generates the signed distance from a set up points
%to a rectangular prism.  This function is used as a distance function
%in DistMesh.
%
%   DRECTANGLE_3D(P,PV)
%      D_SIGNED:     SIGNED DISTANCE TO RECTANGULAR PRISM SURFACE (Nx1)
%      P:            POINTS (NX3)
%      PV:           VERTEX COORDINATES OF RECTANGLULAR PRISM (8X3)
%
%-------------------------------------------------------------------------
%   In order to determine the signed distance, we first decompose
```

```
function [p,t,vert] = QDot_Mesh( QD_fd,B_fd,fh,h0,QD_bbox,B_bbox )

% Create a mesh for the quantum dot material using distmesh ND
[QD_coord,QD_nodes]=distmeshnd(QD_fd,fh, h0,QD_bbox,[]);
[QD_coord,QD_nodes]=fixmesh(QD_coord,QD_nodes);
QD_surface=unique(surftri(QD_coord,QD_nodes));
fixed=[QD_coord(QD_surface(:),1),QD_coord(QD_surface(:),2),QD_coord(QD_surface(:),3)];
% Create a mesh for the barrier material using distmesh ND
%
[B_coord,B_nodes]=distmeshnd(B_fd,fh,h0,B_bbox,fixed);
[B_coord,B_nodes]=fixmesh(B_coord,B_nodes);
B_surface=unique(surftri(B_coord,B_nodes));
% Convert coordinates to desired significant digits
%
B_coord=round(B_coord*1000)/1000;
QD_coord=round(QD_coord*1000)/1000;
fixed=round(fixed*1000)/1000;
vert = fixed;
% 1. Get coordinates of all the nodes on edges of the barrier material
edges=[B_coord(B_surface(:),1),B_coord(B_surface(:),2),B_coord(B_surface(:),3)];
% 2. Start to systematically eliminate nodes on barrier bounding cube
for i=1:2
    for j=1:3
        [r c]=find(edges == B_bbox(i,j));
        edges(r,:)=[];
    end
end
% 3. Start to systematically eliminate nodes associated with nodes on
% the QD surface.
%
[index]=ismember(edges,fixed,'rows');
if (~isempty(index))
edges(index,:)=[];
end
% 4. Eliminate "orphan" nodes
%
[index]=ismember(B_coord,edges,'rows');
B_coord(index,:)=[];
% 5. Complete the final mesh
%
p=unique([B_coord;QD_coord],'rows');
t=delaunayn(p);
%
% 6. View mesh
simpplot(p,t,'p(:,2)>0');
drawnow
end
```

Figure 5.5: The complete source code for generating a three dimensional quantum dot mesh called QDot_mesh.

```
%   the surfaces of the polyhedron into surface triangles.  This is
%   accomplished by composing surface triangles in an NX3 matrix, with N
%   number of surface triangles each containing its 3 vertices.
%
%   Variables:
%       T: MATRIX REPRESENTING THE SURFACE TRIANGLES (12X3)
%       NT: NUMBER OF SURFACE TRIANGLES
%       TP: VERTICE POINTS OF SURFACE TRIANGLE (3X3)
%       DS: DISTANCE OF POINTS FROM SURFACE TRIANGLE USING FUNCTION
%       D: UNSIGNED DISTANCE TO RECTANGULAR PRISM SURFACE (NX1)
%   Functions:
%       POLYGON_CENTROID_3D - calculates the centroid of the polyhedron.
%       TRIANGLE_POINT_DIST_3D - generates the distance from a set of
%           points to a triangle in 3D.
%       INHULL - tests if a set of points is inside a convex hull.
%
T=[1 2 6;1 6 5;2 3 7;2 7 6;3 4 8;3 8 7;1 4 8;1 8 5;1 2 3;1 4 3;5 6 7;5 8 7];
centroid = polygon_centroid_3d(T,pv);
%
%   The distance from each surface triangle is computed and then minimized
%   to determine the distance.
nT=size(T,1);
%
for j=1:nT
    TP = [pv(T(j,1),:);pv(T(j,2),:);pv(T(j,3),:)];
    ds(:,j) = triangle_point_dist_3d(TP,p,centroid);
end

d=min(ds,[],2);
%   The function inhull is used to determine the signed distance.  If the
%   point lies within the polyhedron, the distance is negative.  If
%   the point lies outside the polyhedron, the distance is positive.
d_signed=(-1).^(inhull(p,pv)).*d;
end
```

The function *drectangle_3D* produces the signed distance from the rectangular prism to the points $\mathbf{p}$ (see Fig. 5.6). As *Distmesh* requires, a negative sign will be assigned to points inside and a positive sign will be assigned to points outside. The input arguments are points $\mathbf{p}$ for which the signed distance is computed and 8 vertices of rectangular prism (see Fig. 5.6). After the initial calling sequence, the surface of the polyhedron is decomposed into surface triangles using the vertices of the rectangular prism. There's a total of 12 surface triangles as labeled by the 12×3 matrix $\mathbf{T}$. Distances between the points and each of the surface triangles is computed, then those distances are minimized. The minimized value is the distance to the rectangular prism boundary. The sign of the distance is determined if the point lies inside or outside the rectangular prism. There are two main functions within the code to accomplish these tasks. The first is the function *triangle_point_dist_3D*, which computes the distance from the points to a surface triangle in 3D. The second is *inhull*, which computes whether a point lies inside a polyhedron [58]. *Inhull's* output is a vector that contains 1's and 0's, a 1 if the point lies inside and 0 if the point lies outside. When it is used in the formula `d_signed=(-1).^(inhull(p,pv)).*d`, the signed distance is computed. The function *triangle_point_dist_3D* is shown below in order to fully understand *drectangle_3D*:

```
function dist = triangle_point_dist_3d ( t, p, centroid )
%Triangle_point_dist_3d generates the distance from a set of points
%to a triangle in 3D.
%
%   TRIANGLE_POINT_DIST_3D(T,P)
%       DIST: DISTANCE FROM TRIANGLE (Nx1)
%       T: TRIANGLE VERTICES (3X3)
```

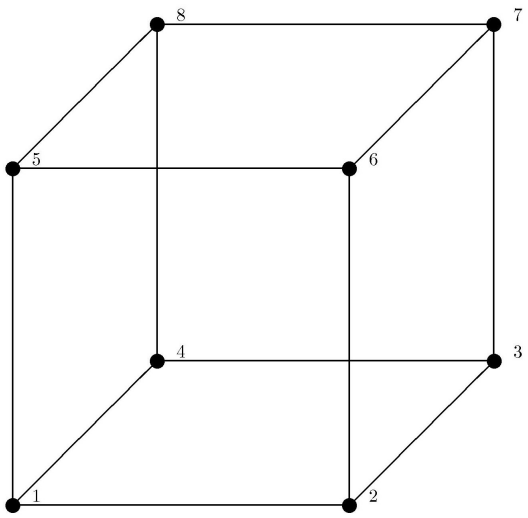


Figure 5.6: A rectangular prism with the vertices labeled 1 through 8. It is made up of 4 tetrahedrons (1,2,3,6), (1,3,4,8), (5,6,8,1), and (6,7,8,3) and 12 surface triangles (1,2,6), (1,6,5), (2,3,7), (2,7,6), (3,4,8), (3,8,7), (1,4,8), (1,8,5), (1,2,3),(1,4,3), (5,6,7), and (5,8,7).

```
%       P: POINTS (NX3)
%       CENTROID: CENTROID OF POLYHEDRON
%
%-------------------------------------------------------------------------
%   This function is used to find the distance of a set of 3D points to a
%   triangle defined by its vertices.
%
%   The algorithm does the following:
%       1.  Determine the distances to each triangle edge and vertex.
%       These distances are minimized.
%       2.  Determine the distances to the plane the triangle lies on.
%       3.  Project the points to the plane on the triangles plane.
%       4.  Determine if the projected points are inside or outside the
%       triangle.
%       5.  For those points inside, the distance is to the plane the
%       triangle lies on.
%
%   Variables:
%       DIST2: DISTANCE FROM POINTS TO SEGMENTS OR VERTICIES IN TRIANGLE
%       DIST3: DISTANCE FROM POINTS TO PLANE IN TRIANGLE
%       P_0: POINTS PROJECTED ON PLANE OF TRIANGLE (NX3)
%       FLAG: NX1 ARRAY USED TO DISTINGUISH IF POINT LIES OUTSIDE OR INSIDE
%           OF TRIANGLE.  IF OUTSIDE, FLAG IS 0.  IF INSIDE, FLAG IS 1 (NX1)
%       IND: INDICES CORRESPONDING TO FLAG EQUAL TO 1.
%   Functions:
%       SEGMENT_POINT_DIST_3D - distance from a set of points
%           to a line segment in 3D.
%       PLANE_VERT_POINT_DIST_3D - computes the distance from a point to
%           plane using the 3 vertices of a triangle.
%       INHULL - tests if a set of points is inside a convex hull.
%
%   Compute the distances from the points to each of the sides or vertex using
%   the function seqment_point_dist_3d.  Minimize these distances.
```

```
%
dist2 = segment_point_dist_3d ( t(1,:), t(2,:), p );
dist = dist2;

dist2 = segment_point_dist_3d( t(2,:), t(3,:), p );
dist = min ( dist, dist2 );

dist2 = segment_point_dist_3d ( t(3,:), t(1,:), p );

dist = min ( dist, dist2 );
%   Compute the distance from the points to the plane using the function
%   plane_vert_point_dist_3d.  This function (plane_vert_point_dist_3d)
%   calls the two functions plane_vert2std_3d and plane_std_point_dist3d
%   which converts to the standard form of the plane and computes the
%   distance while producing a set of projected points respectively.
%   Use of the inhull function determines if the projected points lie
%   inside or outside the triangle.  To do this, we need to feed inhull a
%   fourth point not on the plane.  This allows inhull to determine those
%   points within the triangle (added 4th point creates a tetrahedron).
%   The 4th point can be chosen arbitrarily, so we use its centroid of the
% quantum dot shape.

[dist3,p_0] = plane_vert_point_dist_3d ( t(1,:),t(2,:),t(3,:),p );
dist4 = min ( dist , dist3 );
flag=inhull(p_0,[t; centroid],[],1.0e-05);
ind=find(flag);
% Replace distances
dist(ind) = dist4(ind);
return
end
```

The text within the code above gives a description of the algorithm and subroutine functions. It must be noted that throughout the entire code, including subroutine code, loops are avoided to maximize computational speed. This is especially important in MATLAB because it is an interpretive language.

2. The distance function for the five sided pyramid:

```
function d_signed=dFiveSidedPyramid(p,pv)
%dFiveSidedPyramid generates the signed distance from a set up points
%to a five sided pyramid.  This function is used as a distance function
%in DistMesh.
%
%   DFIVESIDEDPYRAMID(P,PV)
%      D_SIGNED:     SIGNED DISTANCE TO FIVE SIDED PYRAMID SURFACE (Nx1)
%      P:            POINTS (NX3)
%      PV:           VERTEX COORDINATES OF 5-SIDED PYRAMID
%
%-------------------------------------------------------------------------
%   In order to determine the signed distance, we first decompose
%   the surfaces of the polyhedron into surface triangles.  This is
%   accomplished by composing the surface triangles in an NX3 matrix, with N
%   number of surface triangles each containing its 3 vertices.
%
%   Variables:
%       T: MATRIX REPRESENTING THE SURFACE TRIANGLES (6X3)
%       NT: NUMBER OF SURFACE TRIANGLES
%       TP: VERTICE POINTS OF SURFACE TRIANGLE (3X3)
%       DS: DISTANCE OF POINTS FROM SURFACE TRIANGLE USING FUNCTION
```

```
%       D: UNSIGNED DISTANCE TO FIVE SIDED PYRAMID SURFACE (NX1)
%   Functions:
%       POLYGON_CENTROID_3D - calculates the centroid of the polyhedron.
%       TRIANGLE_POINT_DIST_3D - generates the distance from a set of
%           points to a triangle in 3D.
%       INHULL - tests if a set of points are inside a convex hull.
%
T=[1 2 5; 2 3 5; 3 4 5;4 5 1;1 2 4; 2 3 4];
centroid = polygon_centroid_3d(T,pv);
%
%   The distance from each surface triangle is computed and then minimized
%   to determine the distance.
nT=size(T,1);
%
for j=1:nT
    TP = [pv(T(j,1),:);pv(T(j,2),:);pv(T(j,3),:)];
    ds(:,j) = triangle_point_dist_3d(TP,p,centroid);
end

d=min(ds,[],2);
%   The function inhull is used to determine the signed distance.  If the
%   point lies within the polyhedron, the distance is negative.  If
%   the point lies outside the polyhedron, the distance is positive.
d_signed=(-1).^(inhull(p,pv)).*d;
end
```

As expected, this function is similar to the rectangular prism with the only difference in variable **T**. The surface triangles are characteristic for each polyhedron, each matrix **T** identifies a polyhedron.

3. The template for any polyhedron distance function is to modify **T** to account for the surface triangles of a particular polyhedron. As an example consider a tetrahedron as seen in Fig. 5.7. It is composed of the 4 surface triangles (1,2,3), (1,3,4), (2,3,4), and (1,2,4) as represented by its vertices. The matrix **T**= [1 2 3; 1 3 4; 2 3 4; 1 2 4] would be inserted in the code, while the rest of the code is unmodified.

5.2.2 Examples

To further understand how to use **QWire_mesh** and **QDot_mesh**, we show some worked examples.

1. A circular constrained mesh with refinement around the interface. This type of mesh could represent the cross-section of a circular quantum wire embedded in a barrier material. The MATLAB code generating Fig. 5.8b is listed below:

```
>> QW_circle=inline('dcircle(p,0,0,20)','p');
>> B_circle=inline('ddiff(dpoly(p,[-40,-40;40,-40;40,40;-40,40;-40,-40]),
   dcircle(p,0,0,20))','p');
>> [p,t,vert]=QWire_Mesh(QW_circle,B_circle,@fh_finite_2D,5,
   [-20,-20;20,20],[-40,-40;40,40]);
```

The quantum material distance function *QW_circle* is given as an inline function (type 'help inline' for more information). This first argument is the function itself, in this case *dcircle* which is part of the distmesh package and defines a circle with a radius of 20 centered at $(0, 0)$. The second argument, **p**, names the variable to the function. The barrier material distance function *B_circle* is given as an inline function. The function uses *ddiff* in combination with *dpoly* and *dcircle* to remove the circle from a square so that the quantum material is 'cut out' of the barrier material. We use *dpoly* to represent the square by passing its vertices in a 5×2 matrix (notice that the first vertex is included in the last row). These two distance functions are the first two arguments in *QWire_Mesh*. The next argument is a relative edge length function, *fh_finite_2D* that uses *QW_circle* in its definition:

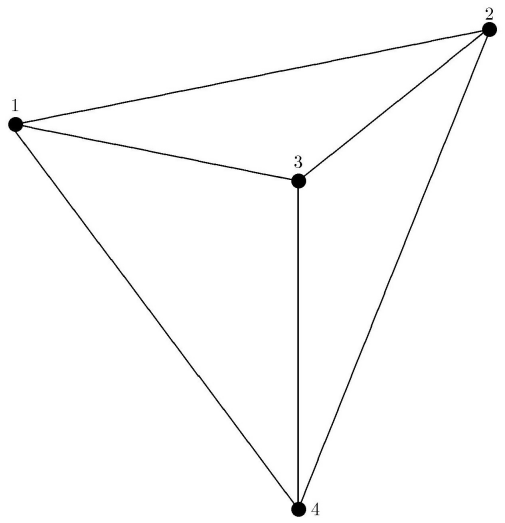


Figure 5.7: A tetrahedron with the vertices labeled 1 through 4. Surface triangles are (1,2,3), (1,3,4), (2,3,4), and (1,2,4).

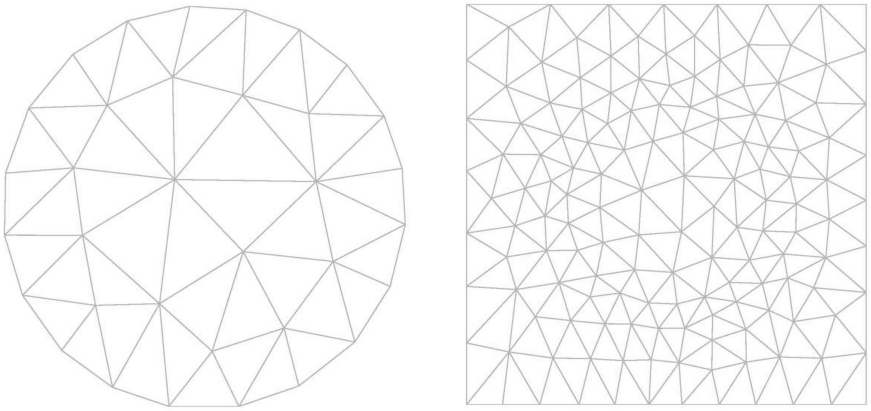
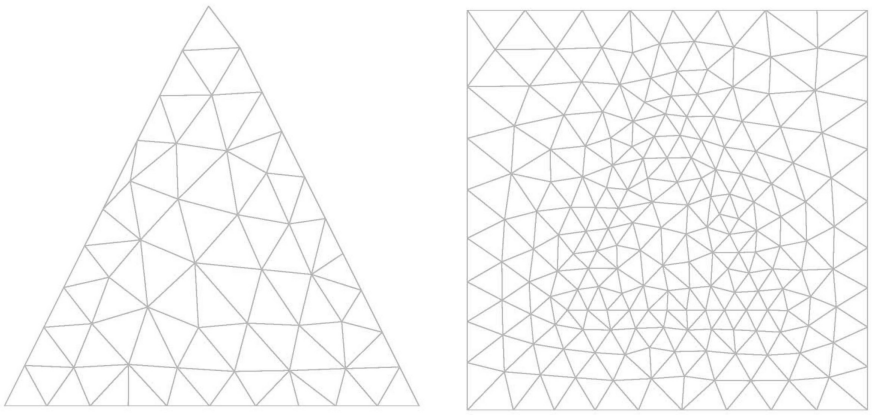


Figure 5.8: (a) Circular mesh (b) A circular constrained mesh with refinement around the interface.

```
function h = fh_finite_2D(p)
%Fh_finite_2D variable edge length that uses a distance function
%   This function is used with QWire_Mesh when it is desired to have more
%   elements near the quantum and barrier material interface.  It uses the
%   distance function of the quantum wire.
%
%   [H]=FH_FINITE_2D(P)
%      H:   EDGE LENGTH
%      P:   NODE POSITION (Nx2)
%------------------------------------------------------------------------
%   Variables:
%       d1: The distance function of the quantum wire.
%       h1: The calibration used in defining the edge lengths, linear
%       function.
% h: Minimum distance, either h1 or scalar.
d1=dcircle(p,0,0,20);
h1=5+0.2*(abs(d1));
h=min(h1,10);
end
```

The element sizes increase with the distance from the interface in **d1** with the approximate maximum edge length being some scalar, in this example 10, of our choosing in **h**. We highly recommend using this type of relative edge length function when modeling quantum wire cross-sections or dots due to its simplicity and ability to produce more elements near the interface relative to the rest of the mesh. The next argument is an initial edge length, 5, and the last two arguments are the bounding boxes of the quantum material and barrier material respectively.

2. A triangular constrained mesh with refinement around the interface. This type of mesh could represent the cross-section of a triangular quantum wire embedded in a barrier material. The MATLAB code generating Fig. 5.9b is listed below:



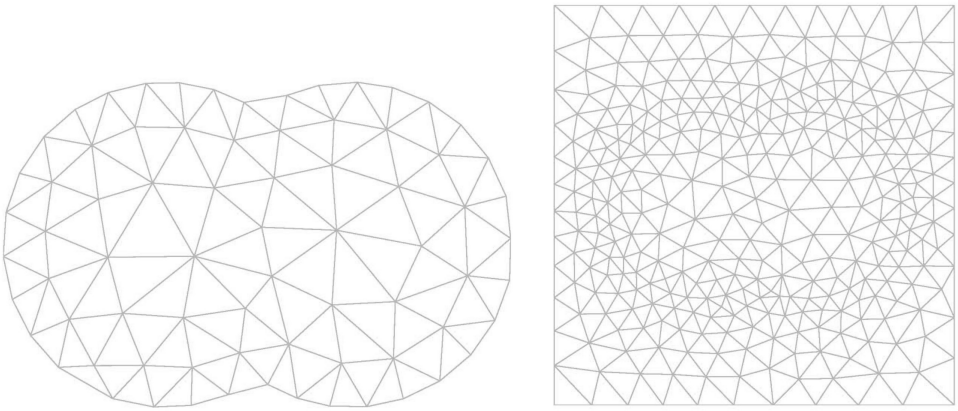
(a) **(b)**

Figure 5.9: (a) Triangular mesh (b) A triangular constrained mesh with refinement around the interface.

```
>> QW_triangle=inline('dpoly(p,[-20,-20;20,-20;0,20;-20,-20])','p');
>> B_triangle=inline('ddiff(dpoly(p,[-40,-40;40,-40;40,40;-40,40;-40,-40]),
   dpoly(p,[-20,-20;20,-20;0,20;-20,-20]))','p');
>> [p,t,vert]=QWire_Mesh( QW_triangle,B_triangle,@fh_finite_2D,4,
   [-20,-20;20,20],[-40,-40;40,40] );
```

The quantum material distance function *QW_triangle* is given as an inline function. This function uses *dpoly* to define a triangle with vertices (-20,-20), (20,20), and (0,20). The barrier material distance function *B_triangle* is given as an inline function. Again, the function uses *ddiff* in combination with *dpoly* to remove the triangle from a square with vertices (-40,-40), (40,-40), (40,40), and (-40,40) so that the quantum material geometry is 'cut out' of the barrier material. These two distance functions are the first two arguments in *QWire_Mesh*. The next argument is the relative edge length function *fh_finite_2D* that uses *QW_triangle* in its definition along with `h1=4+0.2*(abs(d1))` and `h=min(h1,10)`. The element sizes increase with the distance from the interface with the approximate maximum edge length of 10 and minimum edge length of 4. The next argument is the initial edge length of 4 and the last two arguments are the bounding boxes of the quantum material and barrier material respectively.

3. A more complicated constrained mesh with refinement around the interface. The MATLAB code generating Fig. 5.10b is listed below:



(a) **(b)**

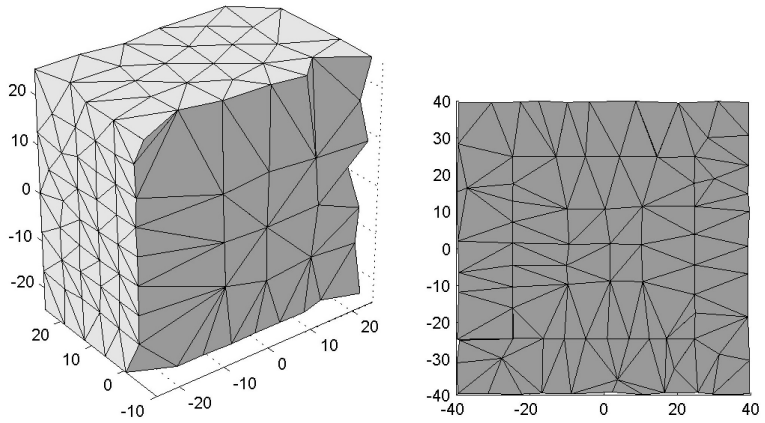
Figure 5.10: (a) More complicated mesh (b) Constrained mesh containing a more complicated mesh.

```
>> QW = inline('dunion(dcircle(p,-20,0,35),dcircle(p,20,0,35))','p');
>> B = inline('ddiff(dpoly(p,[-75,-75;75,-75;75,75;-75,75;-75,-75]),
   dunion(dcircle(p,-20,0,35),dcircle(p,20,0,35)))','p');
>> [p,t,vert]=QWire_Mesh( QW,B,@fh_finite_2D,6,[-50,-50;50,50],
   [-75,-75;75,75] );
```

The quantum material distance function *QW* is given as an inline function. This function uses *dunion* in combination with *dcircle* to combine two circles with radii of 35 and centers located at (-20,0) and (20,0). The barrier material distance function *B* is given as an inline function. This function uses *ddiff* in combination with *dunion* and *dcircle* to remove the complicated geometry from a square with vertices (-75,-75), (75,-75), (75,75), and (-75,75) so that the quantum material is 'cut out' of the barrier material. These two distance functions are the first two arguments in *QWire_Mesh*. The next argument is the relative edge length function *fh_finite_2D* that uses *QW* in its definition along with `h1=6+0.2*(abs(d1))` and `h=min(h1,17)`. The element sizes increase with the distance from the interface with the approximate maximum edge length of 17 and minimum edge length of 6. The next argument is the initial edge length of 6 and the last two arguments are the bounding boxes of the quantum material and barrier material respectively.

4. A rectangular prism constrained mesh representing a quantum dot embedded in barrier material with refinement around the interface. The MATLAB code generating Fig. 5.11b is listed below:

```
>> QD_prism=inline('drectangle_3D(p,[-25,-25,-25;25,-25,-25;25,25,-25;
   -25,25,-25;-25,-25,25;25,-25,25;25,25,25;-25,25,25])','p');
>> B_prism=inline('ddiff(drectangle_3D(p,[-40,-40,-40;40,-40,-40;40,40,-40;
   -40,40,-40;-40,-40,40;40,-40,40;40,40,40;-40,40,40]),drectangle_3D(p,
   [-25,-25,-25;25,-25,-25;25,25,-25;-25,25,-25;-25,-25,25;25,-25,25;
   25,25,25;-25,25,25]))','p');
>> [p t vert] = QDot_Mesh(QD_prism,B_prism,@fh_finite_3D,6,
   [-25,-25,-25;25,25,25],[-40,-40,-40;40,40,40]);
```



(a) **(b)**

Figure 5.11: (a) Cross section of a rectangular prism quantum dot mesh (b) Constrained mesh on $x-z$ plane containing quantum dot embedded in barrier material.

The quantum material distance function *QD_prism* is given as an inline function. This function uses *drectangle_3D* to represent a rectangular prism by passing its vertices in an 8×3 matrix. The barrier material distance function *B_prism* is given as an inline function. This function uses *ddiff* to remove the quantum material geometry from the barrier material. These two distance functions are the first two arguments in *QDot_Mesh*. The next argument is the relative edge length function *fh_finite_3D*, which has all the same definitions and code structure as *fh_finite_2D*, and uses *QW* in its definition along with `h1=6+1*(abs(d1))` and `h=min(h1,12)`. The element sizes increase with the distance from the interface with the approximate maximum edge length of 12 and minimum edge length of 6. The next argument is the initial edge length of 6 and the last two arguments are the bounding boxes of the quantum material and barrier material respectively. In this example, the constrained mesh contains a total of 800 points and 3764 tetrahedrons.

5. A five sided pyramid constrained mesh representing a quantum dot embedded in barrier material with refinement around the interface. The MATLAB code generating Fig. 5.12b is listed below:

```
>> QD_5sided=inline('dFiveSidedPyramid(p,[-25,-25,-25;25,-25,-25;25,25,-25;
   -25,25,-25;0,0,25])','p');
>> B_5sided=inline('ddiff(drectangle_3D(p,[-40,-40,-40;40,-40,-40;40,40,-40;
   -40,40,-40;-40,-40,40;40,-40,40;40,40,40;-40,40,40]),dFiveSidedPyramid(p,
   [-25,-25,-25;25,-25,-25;25,25,-25;-25,25,-25;0,0,25]))','p');
>> [p t vert] = QDot_Mesh(QD_prism,B_prism,@fh_finite_3D,6,
   [-25,-25,-25;25,25,25],[-40,-40,-40;40,40,40]);
```

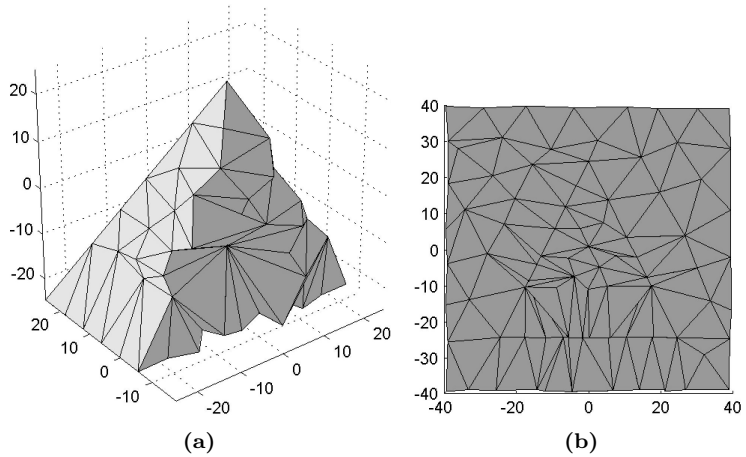


Figure 5.12: (a) Cross section of a five sided pyramid quantum dot mesh (b) constrained mesh on $x - z$ plane containing quantum dot embedded in barrier material.

The quantum material distance function *QD_5sided* is given as an inline function. This function uses *dFiveSidedPyramid* to represent a five sided pyramid by passing its vertices in a 5×3 matrix. The barrier material distance function *B_5sided* is given as an inline function. This function uses *ddiff* to remove the quantum material geometry from the barrier material. These two distance functions are the first two arguments in *QDot_Mesh*. The next argument is the relative edge length function *fh_finite_3D*that uses *QW* in its definition along with `h1=6+1*(abs(d1))` and `h=min(h1,12)`. The element sizes increase with the distance from the interface with the approximate maximum edge length of 12 and minimum edge length of 6. The next argument is the initial edge length of 6 and the last two arguments are the bounding boxes of the quantum material and barrier material respectively. In this example, the constrained mesh contains a total of 556 points and 2620 tetrahedrons.

5.3 FEM Program - Processing Code

As we saw in Chap. 4, FEM requires evaluating integrals and populating matrices that formulate the generalized eigenvalue problem. This is the goal of all processing FEM code. Our processing code is no different, except for the way integrals are treated. Most FEM codes numerically evaluate these integrals due to the presence of the Jacobian using Gaussian quadrature as the preferred technique. This approximation uses a weighted sum of function values at specified points within the integral domain [59]. However, this introduces integration error and increases the number of operations needed to perform the integrals. Therefore instead of approximating the integrals, we have carried out the integrals analytically on a benchmark element and 'hardcoded' the results. In 2D, the benchmark element is the standard triangle with vertices $(0, 0)$, $(1, 0)$, and $(0, 1)$. In 3D, the benchmark element is the standard tetrahedron with vertices $(0, 0, 0)$, $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$. Each element from the mesh is linearly mapped to the benchmarked element to ensure the Jacobian entries and its determinant are constant. The benchmark integral evaluations are used with the aid of the Jacobian to determine the integrals. In essence, the step requiring integral evaluation has been removed and the processing code's only objective is reduced to populating the respective matrices. The ideas following in the next few paragraphs explain the general structure of the processing code and highlight the more important aspects. We will then proceed to a detailed description of the code, line by line, as before.

There are a number of variables that must be set before the processing code begins. The first variable, **ndof**, is used to assign the desired degree(s) of freedom per node and makes the processing code flexible in terms of nodal continuity. The choices available are either 1, 3, or 4 and are

described in the next paragraph. The rest of the variables pertain to the quantum and barrier material: effective masses expressed in the electron rest mass unit, i.e. $m_{eff} = 0.3774 * m_e$, and offset potentials expressed in the electron-volt (eV) unit.

Following from the ideas presented in Chap. 4, interpolation basis functions are either linear or cubic. Linear basis functions are used to ensure continuity of the wavefunction at each node, while cubic basis functions are used to ensure continuity of the wavefunction and its first derivative at each node. The choice to use linear or cubic basis functions depends on the degree of freedom per node. Linear functions are used when one degree of freedom per node is selected, while cubic functions are used when either three for 2D or four for 3D degrees of freedom are selected. They were constructed for the right triangle or tetrahedron using the prescription outlined in Chap. 4. Table 5.1 summarizes the two sets of basis functions for the right triangle used in 2D FEM

Table 5.1: The basis functions that are used in 2D processing code for the right triangle with vertices located at (0,0), (1,0), and (0,1). The basis functions are characterized by the degree of freedom (dof) per node indicating the type of continuity applied. The local coordinate system is (ξ,η).

dof	(0,0)	(1,0)	(0,1)
1	$\phi_1 = 1 - \phi_2 - \phi_3$	$\phi_2 = \xi$	$\phi_3 = \eta$
3	$\phi_1 = 1 - \phi_4 - \phi_7$	$\phi_4 = 3\xi^2 - 2\xi^3$	$\phi_7 = 3\eta^2 - 2\eta^3$
	$\phi_2 = \xi - \phi_4 - \phi_5 - \phi_8$	$\phi_5 = -\xi^2 + \xi^3$	$\phi_8 = \frac{1}{3}\left(\xi\eta + 2\xi\eta^2 - \xi^2\eta\right)$
	$\phi_3 = \eta - \phi_6 - \phi_7 - \phi_9$	$\phi_6 = \frac{1}{3}\left(\xi\eta + 2\xi^2\eta - \xi\eta^2\right)$	$\phi_9 = -\eta^2 + \eta^3$

evaluations. Table 5.2 summarizes the two sets of basis functions for the right tetrahedron used in 3D FEM evaluations. It should be noted that the basis functions for four degrees of freedom have been reduced in complexity by exploiting the extra functions, $\phi_{17-19} = -\frac{1}{3}x_i x_j + \frac{1}{3}x_i^2 x_j + \frac{1}{3}x_i x_j^2$ $(1 < i < j < 3)$, that were generated in the process. We have arrived at the algebraic expressions in Tab. 5.2 after subtracting ϕ_{17}-ϕ_{19} from original basis functions: ϕ_2, ϕ_3, ϕ_4, ϕ_7, ϕ_8, ϕ_{10}, ϕ_{12}, ϕ_{14}, and ϕ_{15}.

Table 5.2: The basis functions that are used in 3D processing code for the right tetrahedron with vertices located at (0,0,0), (1,0,0), (0,1,0), and (0,0,1). The basis functions are characterized by the degree of freedom (dof) per node indicating the type of continuity applied. The local coordinate system is (ξ,η,ζ).

dof	(0,0,0)	(1,0,0)	(0,1,0)	(0,0,1)
1	$\phi_1 = 1 - \phi_2 - \phi_3 - \phi_4$	$\phi_2 = \xi$	$\phi_3 = \eta$	$\phi_4 = \zeta$
4	$\phi_1 = 1 - \phi_5 - \phi_9 - \phi_{13}$	$\phi_5 = 3\xi^2 - 2\xi^3$	$\phi_9 = 3\eta^2 - 2\eta^3$	$\phi_{13} = 3\zeta^2 - 2\zeta^3$
	$\phi_2 = \xi - \phi_5 - \phi_6 - \phi_{10} - \phi_{14}$	$\phi_6 = -\xi^2 + \xi^3$	$\phi_{10} = \xi\eta^2$	$\phi_{14} = \xi\zeta^2$
	$\phi_3 = \eta - \phi_7 - \phi_9 - \phi_{11} - \phi_{15}$	$\phi_7 = \xi^2\eta$	$\phi_{11} = -\eta^2 + \eta^3$	$\phi_{15} = \eta\zeta^2$
	$\phi_4 = \zeta - \phi_8 - \phi_{12} - \phi_{13} - \phi_{16}$	$\phi_8 = \xi^2\zeta$	$\phi_{12} = \eta^2\zeta$	$\phi_{16} = -\zeta^2 + \zeta^3$

Once the variables are set, the processing code begins evaluating each element in the mesh. This entails constructing elemental matrices, i.e. kinetic, potential, and overlap. Each of the matrices is populated using the benchmarked integral results and various properties of the Jacobian matrix. In the case when cubic functions are used and the effective masses between the two materials are different, the interface boundary condition is such that the wavefunction and its inverse mass derivative must be continuous [36]. This condition is captured by multiplying the rows and columns corresponding to the unknown derivative amplitudes on the interface nodes by the appropriate

effective mass ratio after the elemental matrices are constructed.

After each elemental matrix is constructed, it must be properly overlaid in the larger respective global matrix. A map linking the two is needed. The map or index system must take into consideration how unknown amplitudes at each node should be organized. If the following notation is used to describe unknown amplitudes at node i,

$$\psi_{i,0} = \psi \quad \psi_{i,1} = \frac{\partial \psi}{\partial x} \quad \psi_{i,2} = \frac{\partial \psi}{\partial y} \quad \psi_{i,3} = \frac{\partial \psi}{\partial z} \tag{5.3}$$

our processing code wishes to organize these unknowns in the global matrices for one or more than one degree of freedom as follows:

$$\boldsymbol{\psi} = \begin{bmatrix} \psi_{1,0} \\ \psi_{2,0} \\ \psi_{3,0} \\ \psi_{4,0} \\ \psi_{5,0} \\ \psi_{6,0} \\ \psi_{7,0} \\ \psi_{8,0} \\ \vdots \end{bmatrix} \quad \textbf{or} \quad \begin{bmatrix} \psi_{1,0} \\ \psi_{1,1} \\ \psi_{1,2} \\ \psi_{1,3} \\ \psi_{2,0} \\ \psi_{2,1} \\ \psi_{2,2} \\ \psi_{2,3} \\ \vdots \end{bmatrix} \tag{5.4}$$

With the desired global matrix organization outlined in Eg. 5.4, the indexing assigns the system degree of freedom to each node using global node numbers and the degree of freedom per node. The index function, *index_2D*, used in the processing code produces the organization of Eq. 5.4 while avoiding inefficient nested 'for loops':

```
function [ index ] = index_2D( nd,ndof )
%Index_2D Assigns the system degree of freedom to the element node
%   Index is system dof vector which can be used to place the elements
%   associated with element matrices in the global matrices.
%
%   [INDEX]=INDEX_2D(ND,NNEL,NDOF)
%      INDEX: SYSTEM DOF VECTOR
%      ND: VECTOR CONTAINING GLOBAL NODE NUMBERS(1 X 3)
%      NDOF: DEGREE OF FREEDOM PER NODE
%--------------------------------------------------------------------------
%
start=(nd-1)*ndof+1;
if (ndof == 1)
    index = start;
else
    index = [start ; start+1 ; start+2];
    index = index(:)';
end
```

After global matrices are constructed, the only task left to do is to apply the problem's boundary conditions. This normally involves post-treating columns and rows of those nodes located on the bounding box. A problem that involves an infinite potential outside a domain of interest requires the wavefunction to vanish at the domain boundary. An equivalent effect in FEM is to set the rows and columns corresponding to nodal wavefunction amplitudes located on the boundary box to zero. This is done for the kinetic, potential, and overlap matrices with the caveat that the overlap matrix diagonals where this occurs are set to 1. When the generalized eigenvalue problem is solved, there will be n eigenvalues with the value of 0 representing n nodal wavefunction amplitudes falling on the bounding box. These eigenvalues are ignored as they do not represent bound states. In an unbounded domain, as in the case of a finite potential well, the wavefunction should vanish at infinities (square integrable). In FEM, the global matrices do not need to be post-treated for this

type of condition. However after solving for potential bound states, their wavefunctions should be checked to see if the nodal wavefunction amplitudes located on the bounding box 'vanish'. If they do not vanish, then it is possible the bounding box should be extended to accommodate the particular wavefunction of interest.

The source code for the 2D processing code is contained below with a description of every input/output argument, variable, and function preceded by the %.

```
function [ glob_K,glob_V,glob_E,enodes ] = FEM_3D_QD(p,t,vert,V_nodes)
%FEM_2D_QW This is finite element code that populates the global matrices
%needed to solve Schrodinger's equation for a quantum dot (3D).
%
%
%   FEM_3D_QD(P,T,VERT,V_NODES)
%       GLOB_K: KINETIC ENERGY MATRIX (SPARSE MATRIX)
%       GLOB_V: POTENTIAL ENERGY MATRIX (SPARSE MATRIX)
%       GLOB_E: OVERLAP MATRIX (SPARSE MATRIX)
%       ENODES: NODES ON THE BOUNDING BOX (NEEDED FOR POST PROCESSING)
%       P: COORDINATES OF NODES (NNODES X 3)
%       T: TETRAHEDRAL ELEMENTS (NEL X NNEL)
%       VERT: VERTICES OF QUANTUM DOT (M X P,M VERTICES IN P DIMENSIONS)
%       V_NODES: POTENTIAL AT EACH NODE OTHER THAN OFFSET IN eV (NNODES X 1)
%-------------------------------------------------------------------------
%
%   Physical constants:
%       m_e: mass of electron in grams
%       h_bar: plancks constant in kilograms-Angstrom per second
%       q: measure of electron-volt in gram-centimeter squared per second
%       squared
%   Variables:
%       ndof: number of degrees of freedom (dof) per node (1 OR 4)
%       m_eff: effective mass of semiconductor in grams(1 X 2)
%       V_offset: conduction or valence band offset in eV (1 X 2)
%       nnel: number of nodes per element
%       nel: number of elements
%       nnodes: number of nodes
%       interface_nodes: nodes on interface separating the two materials
%       elem_size: size of element matrices
%       r,c,v: row & column indices of respective element matrix that point
%           to value v.
%       nd: local node indexing (1 X NNEL)
%       xcoord/ycoord: local coordinate indexing (1 X NNEL)
%       V: potential energy per node (1 X NNEL)
%       m: effective mass of element
%       J: Jacobian Matrix (2 X 2)
%       Elem_K: elemental kinetic matrix (EDOF X EDOF)
%       Elem_V: elemental potential matrix (EDOF X EDOF)
%       Elem_E: elemental overlap matrix (EDOF X EDOF)
%       index: system degree of freedom assigned to element node(1 X EDOF)
%   Functions:
%       surftri: Find surface triangles from mesh
%       inhull: Tests if a set of points is inside a convex hull
%       jacob_3D: The Jacobian for 2D mapping
%       Elem_matrix_K_3D: Computes the elemental kinetic matrix
%       Elem_matrix_V_3D: Computes the elemental potential matrix
%       Elem_matrix_E_3D: Computes the elemental overlap matrix
%       apply_interface_bc_3D: Applies the boundary conditions on the
%           interface between the two materials if dof > 1.
%       index_3D: Assigns the system dof to the element node
%       assemble: Builds up row & column indices
```

```
%       sparse: assembles global matrices by creating a sparse matrix
%
%ENTER USER INPUT HERE-----------------------------------------------------
ndof = 4;
m_e = 9.109*10^-28;
%(1) represents well and (2) represents barrier
m_eff(1)=m_e*0.009;
m_eff(2)=m_e*0.131;
V_offset(1)=0;
V_offset(2)=2.15;
%END USER INPUT-------------------------------------------------------------
%
h_bar = 1.054*10^-19;
q = 1.602*10^-12;
nnel = 4;
nel=length(t(:,1));
nnode=length(p(:,1));
interface_nodes = find(ismember(p,vert,'rows'));
%
%For bound state energies, which this program is solving for, boundary
%conditions require the wavefunction to fall off at regions approaching
%infinity.  In order to satisfy this, we bound the barrier region at
%"sufficient" distances.
%
enodes = unique(surftri(p,t));
%
elem_size = nnel*ndof*nnel*ndof;
r_K = zeros(nel,elem_size);r_V = zeros(nel,elem_size);r_E = zeros(nel,elem_size);
c_K = zeros(nel,elem_size);c_V = zeros(nel,elem_size);c_E = zeros(nel,elem_size);
v_K = zeros(nel,elem_size);v_V = zeros(nel,elem_size);v_E = zeros(nel,elem_size);
%
%Determine if points are inside, outside, or on QD.  IN is a logical vector
%containing 1's for points inside and on QD surface and 0 for points
%outside.
IN = inhull(p,vert,[],0.01);
%QD_elements is a vector (nel X 1) that contains a sum of each element's
%vertices as represented by 1's and 0's.  If this quantity equals nnodes,
%element lies inside QD.
QD_elements = sum(IN(t),2);
%
for iel=1:nel
    nd = t(iel,:);
    xcoord = p(nd,1);
    ycoord = p(nd,2);
    zcoord = p(nd,3);
    if  (QD_elements(iel) == nnel)
        V(1:nnel) = V_offset(1) + V_nodes(nd);
        m = m_eff(1);
    else
        V(1:nnel) = V_offset(2) + V_nodes(nd);
        m = m_eff(2);
    end
    J = jacob_3D(xcoord,ycoord,zcoord);
    %
    %Construct the elemental matrices by using functions Elem_matrix_K,
    %Elem_matrix_V, and Elem_matrix_E.
    elem_K = ((h_bar)^2/(2*m*q))*Elem_matrix_K_3D(J,ndof);
    elem_V = Elem_matrix_V_3D(J,V,ndof);
    elem_E = Elem_matrix_E_3D(J,ndof);
```

```
    %
    %Add reciprical mass condition at interface if there is more than 1
    %dof per node.
    if (ndof > 1 && m == m_eff(1))
        elem_K = apply_interface_bc_3D(elem_K,interface_nodes,nd,m_eff(2)/m);
        elem_V = apply_interface_bc_3D(elem_V,interface_nodes,nd,m_eff(2)/m);
        elem_E = apply_interface_bc_3D(elem_E,interface_nodes,nd,m_eff(2)/m);
    end
    %
    %Builds up row & column indices using assemble function
    %
    index = index_3D(nd,ndof);
    [r_K(iel,:) c_K(iel,:) v_K(iel,:)] = assemble(elem_K,index);
    [r_V(iel,:) c_V(iel,:) v_V(iel,:)] = assemble(elem_V,index);
    [r_E(iel,:) c_E(iel,:) v_E(iel,:)] = assemble(elem_E,index);
end
%
%Construct the global matrices using sparse and triples of
%rows/columns/values
%
glob_K = sparse(r_K,c_K,v_K,ndof*nnode,ndof*nnode);
glob_V = sparse(r_V,c_V,v_V,ndof*nnode,ndof*nnode);
glob_E = sparse(r_E,c_E,v_E,ndof*nnode,ndof*nnode);
%
end
```

The first line of the code specifies the calling sequence syntax for the function FEM_2D_QW:

```
function [ glob_K,glob_V,glob_E,enodes ] = FEM_2D_QW(p,t,vert,V_nodes)
```

This function produces the following output:

- The global kinetic matrix **glob_K**. This is a sparse matrix containing only nonzero values of the kinetic matrix elements.
- The global potential matrix **glob_V**. This is a sparse matrix containing only nonzero values of the potential matrix elements.
- The global overlap matrix **glob_E**. This is a sparse matrix containing only nonzero values of the overlap matrix elements.
- Index vector **enodes** containing nodes on the bounding box, where the wavefunction should fall to zero.

The input arguments are the following:

- The node positions **p**. This $N \times 2$ matrix contains the x and y positions for each of the N nodes.
- The triangle vertices **t**. This $NT \times 3$ matrix contains the 3 vertices, each referring to the respective node number, for each of the NT triangles in the mesh.
- The nodes on the interface **vert**. This vector contains the nodes that fall on the interface separating the two materials.
- The external potential values on each node **V_nodes**. This $N \times 1$ vector contains values of an additional potential besides the offset potential values occurring in the Hamiltonian at each node. An example of an additional potential associated with each node is a strain potential as calculated using FEM, outside of this code. If no other potential is present besides the offset potential, supply the 'zeros' vector using `zeros(length(p),1)`. The units are in eV.

In the beginning of the code, the variables **ndof**, **m_eff**, and **V_offset** are set by the user in the specified units. It is recommended that variable **V_offset(1)**, which refers to the value of the potential in the well (quantum material), be kept at 0 and the value of the offset potential barrier be set in variable **V_offset(2)** for convenience. These are the only variables that need to be defined before the processing code is called.

Following the user input, the code begins by defining the relevant constants in the proper units and assigns values to static variables:

- The number of nodes per element **nnel**, a scalar value.
- The number of elements in mesh **nel**, a scalar value.
- The number of nodes in mesh **nnodes**, a scalar value.
- The nodes on the interface between two materials **interface_nodes**, a vector.
- The number of elements contained in the element matrix **elem_size**, a scalar value.

The next few lines of the code pre-allocate space for indices (**r**,**c**) that point to each element matrix value (**v**) used to eventually build each of the global matrices using the function *sparse*. Next, the logical vector **IN** uses the function *inhull* to determine what nodes lie inside, outside, and on the surface of the quantum well. This information helps determine if an element is inside or outside the quantum material. At this point, the code begins the main 'for loop' and scans through each triangular element. The element's node numbers and coordinates are stored in a local node index **nd** and local coordinate index **xcoord** and **ycoord**. Following the local indexing, the element is determined to lie either inside or outside the quantum material. If the element falls within the quantum material, the appropriate effective mass and offset potential are assigned, if the element falls outside of the quantum material, the barrier effective mass and offset potential are assigned.

The 2×2 Jacobian is determined by calling the function *jacob_2D* and uses linear basis functions as the coordinate transformation. *Jacob_2D* is defined below:

```
function [ J ] = jacob_2D( x,y )
%Jacob_2D The Jacobian for the 2D mapping
%   The Jacobian is constant due to the linear transformation of local
%   coordinates to global coordinates.  Linear basis functions are used as
%   the coordinate transformation.
%
%   [J]=JACOB_2D(X,Y)
%      J: JACOBIAN MATRIX (2x2)
%      X: GLOBAL COORDINATES IN THE X DIRECTION(1x3)
%      Y: GLOBAL COORDINATES IN THE Y DIRECTION(1x3)
%-------------------------------------------------------------------------
J(1,1) = x(2)-x(1);
J(1,2) = y(2)-y(1);
J(2,1) = x(3)-x(1);
J(2,2) = y(3)-y(1);
end
```

The next three lines of the code each use the functions *Elem_matrix_K*, *Elem_matrix_V*, and *Elem_matrix_E* to construct the elemental kinetic, potential, and overlap matrices respectively. The input arguments for the kinetic matrix are variables **J** and **ndof**, while the multiplying prefactor `((h_bar)^2/(2*m*q))` refers to constants associated with kinetic energy. Depending on the degree of freedom, the elemental kinetic energy matrix is either a 3×3 or 9×9 matrix and constructed using the quadratic form as represented by the equation $\frac{\hbar^2}{2m}|J|\psi_i \mathrm{Tr}\left(\mathbf{P_{ij}} \cdot \left(\mathbf{J} \cdot \mathbf{J^t}\right)^{-1}\right)\psi_j$ (see Eq. 4.66), where the trace is over 2×2 matrices. Each matrix $\mathbf{P_{ij}}$ is constructed using four 3×3 or 9×9 matrices $\mathbf{M}_{\mu,\nu}$, with $\mu = (\xi, \eta)$ and $\nu = (\xi, \eta)$. The matrix elements are completed as integrals consisting of a product of basis function derivatives with respect to local coordinates ξ, η.

$$\mathbf{M}_{\mu,\nu}(i,j) = \int_0^1 \int_0^{1-\eta} \nabla_\mu \phi_i(\xi,\eta) \cdot \nabla_\nu \phi_j(\xi,\eta)\, d\xi\, d\eta \quad 1 \leq k,l \leq 3 \text{ or } 9 \tag{5.5}$$

If there is more than one degree of freedom per node, an additional step is needed to remove the unknown derivative amplitude's dependence on the local coordinates. The matrix **translate** does just this:

$$\left[\begin{array}{cc|cc|cc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & J & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & J & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & J \end{array}\right] \tag{5.6}$$

The input arguments for the potential matrix are variables **J**, **V**, and **ndof**. Similar to the kinetic matrix, the potential matrix is either a 3×3 or 9×9 matrix and constructed from the following equation:

$$|J| \int_0^1 \int_0^{1-\eta} \phi_i(\xi,\eta)\, V(x,y)\, \phi_j(\xi,\eta)\, d\xi\, d\eta \quad 1 \le i,j \le 3 \,\text{or}\, 9 \tag{5.7}$$

The integrals are carried out over a piecewise linear approximation to the potential $V(x,y)$ using linear basis functions from Tab. 5.1 and the values of the potential at each of the 3 nodes given in vector **V**:

$$V(x,y) \rightarrow \mathbf{V(1)}\, \phi_1(\xi,\eta) + \mathbf{V(2)}\, \phi_2(\xi,\eta) + \mathbf{V(3)}\, \phi_3(\xi,\eta) \tag{5.8}$$

Again if there is more than one degree of freedom per node, the matrix **translate** is needed to remove the unknown derivative amplitude's dependence on the local coordinates.

The input arguments for the overlap matrix are variables **J** and **ndof**. Similar to the kinetic and potential matrix, the overlap matrix is either a 3×3 or 9×9 matrix and constructed from the following equation:

$$|J| \int_0^1 \int_0^{1-\eta} \phi_i(\xi,\eta)\ \phi_j(\xi,\eta)\, d\xi\, d\eta \quad 1 \le i,j \le 3 \,\text{or}\, 9 \tag{5.9}$$

Again if there is more than one degree of freedom per node, the matrix **translate** is needed to remove the unknown derivative amplitude's dependence on the local coordinates.

After the elemental matrices are constructed, the function *apply_interface_bc_2D* is called if **ndof** > 1 and the element is part of the quantum material. This function determines if any of the nodes fall on the interface and for those that do, post treats the matrix by enforcing the interface boundary condition. This consists of multiplying each respective row and column by the appropriate effective mass ratio.

The elemental matrices now have to be properly placed in each respective global matrix. The function *index_2D* is called to assign the system degree of freedom to each element node and used as an input argument to the function *assemble*. Instead of building up the global matrices from the aggregate one piece at a time, *assemble* stores the row, column, and value information of each respective element matrix in the $nel \times elem\_size$ matrices **r**, **c**, and **v**. We do this to take advantage of the function *sparse*, which only retains nonzero matrix elements. After the code finishes scanning through all the elements, we create the global matrices by calling *sparse*. *Sparse* uses **r**, **c**, and **v** to generate a sparse matrix **S** such that **S(r(k),c(k))** = **v(k)**. Any elements of **v** that are zero are ignored and any elements of **v** that have duplicate values of **r** and **c** are added together.

The source code for the 3D processing code is contained below with a description of every input/output argument, variable, and function preceded by the %. As expected, the source code is similar to the 2D processing with each line of code performing those same functions as expressed above. The steps are synchronized to those steps in the 2D processing code so that a detailed explanation of each line is not necessary.

```
function [ glob_K,glob_V,glob_E,enodes ] = FEM_3D_QD(p,t,vert,V_nodes)
```

```
%FEM_2D_QW This is finite element code that populates the global matrices
%needed to solve Schrodinger's equation for a quantum dot (3D).
%
%
%   FEM_3D_QD(P,T,VERT,V_NODES)
%       GLOB_K: KINETIC ENERGY MATRIX (SPARSE MATRIX)
%       GLOB_V: POTENTIAL ENERGY MATRIX (SPARSE MATRIX)
%       GLOB_E: OVERLAP MATRIX (SPARSE MATRIX)
%       ENODES: NODES ON THE BOUNDING BOX (NEEDED FOR POST PROCESSING)
%       P: COORDINATES OF NODES (NNODES X 3)
%       T: TETRAHEDRAL ELEMENTS (NEL X NNEL)
%       VERT: VERTICES OF QUANTUM DOT (M X P,M VERTICES IN P DIMENSIONS)
%       V_NODES: POTENTIAL AT EACH NODE OTHER THAN OFFSET IN eV (NNODES X 1)
%-------------------------------------------------------------------------
%
%   Physical constants:
%       m_e: mass of electron in grams
%       h_bar: plancks constant in kilograms-Angstrom per second
%       q: measure of electron-volt in gram-centimeter squared per second
%       squared
%   Variables:
%       ndof: number of degrees of freedom (dof) per node (1 OR 4)
%       m_eff: effective mass of semiconductor in grams(1 X 2)
%       V_offset: conduction or valence band offset in eV (1 X 2)
%       nnel: number of nodes per element
%       nel: number of elements
%       nnodes: number of nodes
%       interface_nodes: nodes on interface separating the two materials
%       elem_size: size of element matrices
%       r,c,v: row & column indices of respective element matrix that point
%           to value v.
%       nd: local node indexing (1 X NNEL)
%       xcoord/ycoord: local coordinate indexing (1 X NNEL)
%       V: potential energy per node (1 X NNEL)
%       m: effective mass of element
%       J: Jacobian Matrix (2 X 2)
%       Elem_K: elemental kinetic matrix (EDOF X EDOF)
%       Elem_V: elemental potential matrix (EDOF X EDOF)
%       Elem_E: elemental overlap matrix (EDOF X EDOF)
%       index: system degree of freedom assigned to element node(1 X EDOF)
%   Functions:
%       surftri: Find surface triangles from mesh
%       inhull: Tests if a set of points is inside a convex hull
%       jacob_3D: The Jacobian for 2D mapping
%       Elem_matrix_K_3D: Computes the elemental kinetic matrix
%       Elem_matrix_V_3D: Computes the elemental potential matrix
%       Elem_matrix_E_3D: Computes the elemental overlap matrix
%       apply_interface_bc_3D: Applies the boundary conditions on the
%           interface between the two materials if dof > 1.
%       index_3D: Assigns the system dof to the element node
%       assemble: Builds up row & column indices
%       sparse: assembles global matrices by creating a sparse matrix
%
%ENTER USER INPUT HERE-----------------------------------------------------
ndof = 4;
m_e = 9.109*10^-28;
%(1) represents well and (2) represents barrier
m_eff(1)=m_e*0.009;
m_eff(2)=m_e*0.131;
```

```
V_offset(1)=0;
V_offset(2)=2.15;
%END USER INPUT---------------------------------------------------------------
%
h_bar = 1.054*10^-19;
q = 1.602*10^-12;
nnel = 4;
nel=length(t(:,1));
nnode=length(p(:,1));
interface_nodes = find(ismember(p,vert,'rows'));
%
%For bound state energies, which this program is solving for, boundary
%conditions require the wavefunction to fall off at regions approaching
%infinity.  In order to satisfy this, we bound the barrier region at
%"sufficient" distances.
%
enodes = unique(surftri(p,t));
%
elem_size = nnel*ndof*nnel*ndof;
r_K = zeros(nel,elem_size);r_V = zeros(nel,elem_size);r_E = zeros(nel,elem_size);
c_K = zeros(nel,elem_size);c_V = zeros(nel,elem_size);c_E = zeros(nel,elem_size);
v_K = zeros(nel,elem_size);v_V = zeros(nel,elem_size);v_E = zeros(nel,elem_size);
%
%Determine if points are inside, outside, or on QD.  IN is a logical vector
%containing 1's for points inside and on QD surface and 0 for points
%outside.
IN = inhull(p,vert,[],0.01);
%QD_elements is a vector (nel X 1) that contains a sum of each element's
%vertices as represented by 1's and 0's.  If this quantity equals nnodes,
%element lies inside QD.
QD_elements = sum(IN(t),2);
%
for iel=1:nel
    nd = t(iel,:);
    xcoord = p(nd,1);
    ycoord = p(nd,2);
    zcoord = p(nd,3);
    if  (QD_elements(iel) == nnel)
        V(1:nnel) = V_offset(1) + V_nodes(nd);
        m = m_eff(1);
    else
        V(1:nnel) = V_offset(2) + V_nodes(nd);
        m = m_eff(2);
    end
    J = jacob_3D(xcoord,ycoord,zcoord);
    %
    %Construct the elemental matrices by using functions Elem_matrix_K,
    %Elem_matrix_V, and Elem_matrix_E.
    elem_K = ((h_bar)^2/(2*m*q))*Elem_matrix_K_3D(J,ndof);
    elem_V = Elem_matrix_V_3D(J,V,ndof);
    elem_E = Elem_matrix_E_3D(J,ndof);
    %
    %Add reciprical mass condition at interface if there is more than 1
    %dof per node.
    if (ndof > 1 && m == m_eff(1))
        elem_K = apply_interface_bc_3D(elem_K,interface_nodes,nd,m_eff(2)/m);
        elem_V = apply_interface_bc_3D(elem_V,interface_nodes,nd,m_eff(2)/m);
        elem_E = apply_interface_bc_3D(elem_E,interface_nodes,nd,m_eff(2)/m);
    end
```

```
    %
    %Builds up row & column indices using assemble function
    %
    index = index_3D(nd,ndof);
    [r_K(iel,:) c_K(iel,:) v_K(iel,:)] = assemble(elem_K,index);
    [r_V(iel,:) c_V(iel,:) v_V(iel,:)] = assemble(elem_V,index);
    [r_E(iel,:) c_E(iel,:) v_E(iel,:)] = assemble(elem_E,index);
end
%
%Construct the global matrices using sparse and triples of
%rows/columns/values
%
glob_K = sparse(r_K,c_K,v_K,ndof*nnode,ndof*nnode);
glob_V = sparse(r_V,c_V,v_V,ndof*nnode,ndof*nnode);
glob_E = sparse(r_E,c_E,v_E,ndof*nnode,ndof*nnode);
%
end
```

5.4 FEM Program - Post Processing Code

Schödinger's equation has been turned into three matrices by the processing code that represent the following equation:

$$(H)_{ij}\psi_j = E_{ij}\psi_j \tag{5.10}$$

This equation is in generalized eigenvalue form and the problem is reduced to finding the eigenvalues and eigenvectors. Modern eigensolvers make quick work of this problem by employing optimized algorithms. MATLAB's built in function *sptarn* can solve the generalized eigenvalue problem $Av = \lambda Bv$ using *sparse* matrices to produce sorted generalized eigenvalues λ and a full matrix v whose columns are the corresponding eigenvectors. As such, this function is the pillar in the post processing code that generates the solution to Eq. 5.10. We will now begin with a detailed description of the post processing code.

The source code for the 2D post processing code is contained below with a description of every input/output argument, variable, and function preceded by the %.

```
function [ E psi ] = Energy_psi( glob_K,glob_V,glob_E,ndof,V )
%Energy_psi Computes the energy eigenvalues and wavefuntion
%   Using the global matricies obtained from processing code, this function
%   will compute the energy and normalized wavefunction.  The energy levels
%   are sorted from lowest to highest.
%
%   [E PSI]=ENERGY_PSI ( GLOB_K,GLOB_V,GLOB_E,NDOF,V )
%      E: ENERGY EIGENVALUES
%      PSI: NORMALIZED WAVEFUNCTION
%      GLOB_K: GLOBAL KINETIC ENERGY MATRIX
%      GLOB_V: GLOBAL POTENTIAL ENERGY MATRIX
%      GLOB_E: GLOBAL OVERLAP MATRIX
%      NDOF:  DEGREES OF FREEDOM PER NODE
%      V: VALUE OF SMALLEST POTENTIAL VALUE IN MESH
%-------------------------------------------------------------------------
%   Functions:
%      SPTARN: Eigenvalues and eigenvectors of sparse matrix in interval
%      [0.1*V,V].

[psi E result] = sptarn(glob_K+glob_V,glob_E,0.1*V,V);
psi = psi(1:ndof:end,:);
for i=1:length(psi(1,:))
    normal = norm(psi(:,i));
```

```
    psi(:,i) = psi(:,i)./normal;
end
end
```

The first line of the code specifies the calling sequence of the syntax for the function Energy_psi:

```
function [ E psi ] = Energy_psi( glob_K,glob_V,glob_E,ndof,V )
```

This function produces the following output:

- The sorted eigenvalues **E**. This vector contains the eigenvalues within the specified interval $(0.1 * V, V)$, where V is the value of the smallest potential value in the mesh.
- The sorted eigenvectors **psi**. This matrix contains the wavefunction values at each node in the columns.

The input arguments are the following:

- Sparse global kinetic matrix **glob_K**. This sparse matrix contains the kinetic matrix elements.
- Sparse global potential matrix **glob_V**. This sparse matrix contains the potential matrix elements.
- Sparse global overlap matrix **glob_E**. This sparse matrix contains the overlap matrix elements.
- The degrees of freedom per node **ndof**.
- The value of the smallest potential value in the mesh **V**.

In the beginning of the code, the function *sptarn* is called with input arguments **glob_K**, **glob_V**, **glob_E**, $0.1 * V$, and V, which solves for the eigenvalues and eigenvectors of Eq. 5.10 in the interval $0.1 * V \leq \mathbf{E} \leq V$. The next line in the code 'fixes' the eigenvectors by including only wavefunction amplitudes at each node (wavefunction derivative amplitudes resulting from **ndof** > 1 are discarded). Finally, the wavefunction's corresponding to the eigenvalues are normalized.

Only those eigenvalues with values less than the smallest potential energy value at a particular node may result in bound states. Their wavefunction should be checked to see if it satisfies the square integrable boundary condition. A useful test is to divide the nodes into two groups: those on the bounding box into α and those not into β. If the ratio of $\sum_\alpha |\psi_\alpha|^2 / \sum_\beta |\psi_\beta|^2$ is small, then the solutions might be of some interest and carry some physical meaning. It is important to recognize a number of eigenvalues will be meaningless and are only an artifact of the diagonalization. The following piece of code can produce the previously described test for eigenvalue n:

```
>> A = psi(enodes,n)'*psi(enodes,n);
>> int_nodes = 1:length(p);
>> int_nodes(enodes)=[];
>> B = psi(int_nodes,n)'*psi(int_nodes,n);
>> A/B
```

It is sometimes desirable to visualize the wavefunction to better understand its attributes. In 2D, it is easy to build a 3D plot from the coordinates of the nodes, (x_i, y_i), and the wavefunction values, ψ_i, at those points through an interpolation. The following function *simp_psi_plot* performs a 3D plot of the wavefunction using the triples of values (x_i, y_i, ψ_i) obtained from *Energy_psi*:

```
function simp_psi_plot( gcoord,psi,n )
%simp_psi_plot Plots the wavefuntion corresponding to the energy
%eigvenvalue number obtained from Energy_psi.
%
%   SIMP_PSI_PLOT ( GCOORD,PSI,N )
%      GCOORD: COORDINATES OF MESH
%      PSI: WAVEFUNCTION
%      N: ENERGY LEVEL NUMBER (OBTAINED FROM ENERGY_PSI)
%------------------------------------------------------------------------
```

```
%   Variables:
%       F: interpolated function defined by coordinates of mesh corresponding
%       to the values of the wavefunction.
%       i: increment in x direction
%       j: increment in y direction
%       x: x coordinates used in the surface plot
%       y: y coordinates used in the surface plot
%       Z: interpolated wavefunction values at x and y coordinates
%   Functions:
%       TriScatteredInterp: Interpolate scattered data
%       Meshgrid: Generate X and Y  arrays for 3D plots
%       Surf: 3D shaded surface plot
F = TriScatteredInterp(gcoord(:,1),gcoord(:,2),psi(:,n));
i = (max(gcoord(:,1))-min(gcoord(:,1)))/50;
j = (max(gcoord(:,2))-min(gcoord(:,2)))/50;
x = min(gcoord(:,1)):i:max(gcoord(:,1));
y = min(gcoord(:,2)):j:max(gcoord(:,2));
[X Y] = meshgrid(x,y);
Z = F(X,Y);
surf(X,Y,Z);
end
```

The function begins by calling the MATLAB function *TriScatteredInterp*, which produces an interpolated function using the triples (x_i, y_i, ψ_i). The next few lines define the points on the physical grid where the wavefunction is to be evaluated from the interpolated function **F**. The function *surfc* displays the interpolated wavefunction and draws a contour plot beneath the surface.

The source code for the 3D post processing code is the same as the 2D processing. However, we have not included a visualization function for the wavefunctions due to the complexity arising when trying to project four dimensions on a computer screen.

5.5 FEM Program - Benchmarked Solutions

It is important for any computer program that has the purpose of solving physical problems to be benchmarked against known solutions to test its accuracy and determine any inherent limitations. Problems for the comparison should either come from known analytical or published solutions. In this section, we have benchmarked our FEM program against a few known solutions in 2D and 3D.

5.5.1 Circular symmetric finite potential well in 2D

We begin the benchmarking with a circular symmetric finite potential well. A particle of mass m is confined to the potential:

$$V(r) = \left\{ \begin{array}{ll} 0, & 0 \leq r < a \\ V_0, & r > a \end{array} \right\} \tag{5.11}$$

where V_0 is a constant potential. Schödinger's equation in polar coordinates is:

$$\frac{-\hbar^2}{2m}\left(\frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2}\right)\psi(r,\theta) + V(r)\psi(r,\theta) = E\psi(r,\theta) \tag{5.12}$$

Energy levels in a spherical circular well are found by factoring the wavefunction into a radial and angular part. The angular dependence has the form $\psi(\theta) \simeq e^{i2\pi m\theta}$, where m must be an integer. The radial equation is just the Bessel differential equation in the interior of the well and modified Bessel differential equation outside the well [59]. Solutions are subject to the boundary conditions:

$$R(r) \text{ is finite as } \quad r \to 0 \quad \text{and} \quad R(r) \to 0 \quad \text{as} \quad r \to \infty \tag{5.13}$$

In the interior of the well the solutions are Bessel functions $R(r) \simeq J_m(kr)$ of the first kind, with $E = \hbar^2 k^2/2m > 0$. The second solutions (Y_m) are rejected because they diverge and not square integrable at the origin. For $r > a$ and $E < V_0$, the solutions are modified Bessel functions $R(r) \simeq K_m(\kappa r)$ of the second kind, with $E - V_0 = \hbar^2(\kappa)^2/2m < 0$. We use the solution that decreases exponentially to zero as $r \to \infty$. The solutions are determined by enforcing continuity at $r = a$:

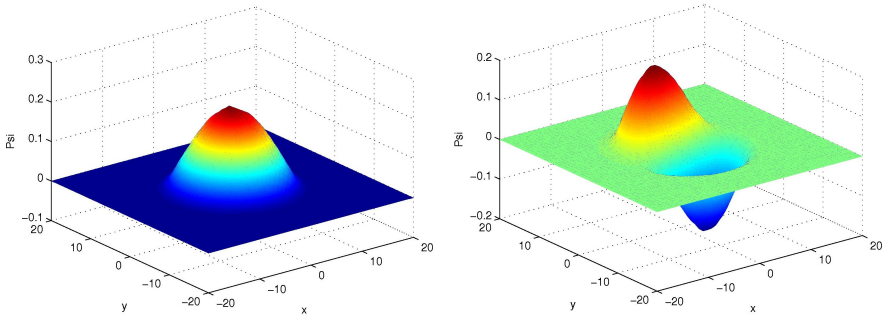
$$R_{\text{inside}}(r)|_{r=a} = R_{\text{outside}}(r)|_{r=a} \quad \text{and} \quad \frac{R_{\text{inside}}(r)}{dr}|_{r=a} = \frac{R_{\text{outside}}(r)}{dr}|_{r=a} \tag{5.14}$$

The bound state energy eigenvalues are labeled $E_{n,m}$, where n is the number of radial nodes and m is the orbital angular momentum. The first five bound state energies found using the outlined approach are compared with an FEM calculation for the circular potential well with a radius $a = 10$ Å and are included in Tab. 5.3. The generated mesh used in the calculations consisted of 480 nodes and 913 elements. The main observation from Tab. 5.3 is that cubic basis functions are very

Table 5.3: The first five bound state energy eigenvalues of the finite circular potential well with $a = 10$Å and $V_0 = 5$ eV using the FEM program with linear and cubic basis functions. The approximation was carried out using 913 elements.

$E_{n,m}$	Energy eigenvalues (eV) linear basis	Energy eigenvalues (eV) cubic basis	Exact eigenvalues (eV)
(0,0)	0.190	0.186	0.186
(1,1)	0.489	0.473	0.472
(2,2)	0.899	0.848	0.846
(1,0)	1.06	0.979	0.976
(3,2)	1.421	1.307	1.304

accurate and both sets of FEM calculations yield a more accurate solution for the lower energy levels. The wavefunction for the ground state and first excited state obtained from the FEM calculation are displayed in Fig. 5.13a and 5.13a below.



(a) **(b)**

Figure 5.13: (a) Wavefunction for the ground state of circular potential well; (b) Wavefunction for the first excited state of circular potential well.

5.5.2 Quantum wire square cross-section

In this example, we consider a quantum wire of square or rectangular cross section with a confining potential arising from the band offset surrounding the material. The offset potential $V(x, y)$ is

non-separable and characterized by:

$$V(r) = \left\{ \begin{array}{ll} 0, & -a/2 \leq x \leq a/2 \text{ and } -b/2 \leq y \leq b/2 \\ V_0, & \text{elsewhere} \end{array} \right\} \tag{5.15}$$

where the dimensions of the quantum wire are $a \times b$. The effective mass approximation is used in Schödinger's equation:

$$\frac{-\hbar^2}{2m^*}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\psi(x,y) + V(x,y)\psi(x,y) = E\psi(x,y) \tag{5.16}$$

where m^* is the either the effective mass of the quantum material m_W or barrier material m_B. Ram-Mohan solved this problem for the rectangular and square quantum wires of GaAs/$Al_{0.37}Ga_{0.63}$As using FEM for energy levels in the conduction band offset [46]. The bound state energies are compared with our FEM calculation for the parameters included in Tab. 5.4. Similar to the circular

Table 5.4: Conduction band energy levels of a GaAs/$Al_{0.37}Ga_{0.63}$As quantum wire with square and rectangular cross section. The parameters $m_W = 0.0665 \cdot m_e$, $m_B = 0.0858 \cdot m_e$, and $V_0 = 0.276$ eV are used with linear and cubic basis functions to compare against benchmark solution.

Cross-sectional area $a \times b$ (Å^2)	Energy eigenvalues (eV) linear basis	Energy eigenvalues (eV) cubic basis	Energy eigenvalues (eV) Benchmark
50 × 50	0.1570	0.1557	0.1553
100 × 50	0.1129	0.1113	0.1111
	0.2017	0.1988	0.1976
100 × 100	0.0658	0.0634	0.0635
	0.1627	0.1550	0.1552
	0.1628	0.1551	0.1552
	0.2518	0.2393	0.2396
	0.2815	0.2745	0.2742

potential well, cubic basis functions are more accurate than linear basis functions and converge to benchmarked solutions within 0.0001 eV. We must note an interesting observation regarding the 100 × 100 square well and the least bound energy level 0.2742 eV. Its wavefunction extends deep into the barrier region due to small $V - E$. Therefore to capture this bound state, the bounding box's dimensions were extended to accompany this wavefunction's penetration. Typically, we start with a bounding box that has the dimensions 2 times the largest quantum material's spacial dimension. However, to accurately evaluate less tightly bound states, the bounding box must be extended. In this case, the bounding box was extended to 400 × 400 Å^2.

5.5.3 Spherically symmetric finite potential well in 3D

Now we benchmark the FEM program in 3D by considering a spherically symmetric finite potential and a spherical quantum dot in the effective mass approximation. A particle of m is confined to the potential given in Eq. 5.11. Schödinger's equation is expressed in spherical coordinates. Energy levels are found by factoring the wavefunction into a radial and angular part $(R_{nl}(r)Y_m^l(\theta,\phi))$ and solving the radial equation subject to the boundary conditions in Eq. 5.13. In the interior of the well the solutions are spherical Bessel functions $R(r) \simeq j_n(kr)$ of the first kind, with $E = \hbar^2k^2/2m > 0$. The second solutions (y_n) are rejected because they diverge and are not square integrable at the origin. For $r > a$ and $E < V_0$, the solutions are modified spherical Bessel functions $R(r) \simeq k_n(\kappa r)$ of

the second kind, with $V_0 - E = \hbar^2(\kappa)^2/2m < 0$. We use the solution that decreases exponentially to zero as $r \to \infty$. The solutions are determined by enforcing continuity at $r = a$, Eq. 5.14. Similarly to the circular potential well, bound state energy eigenvalues are labeled $E_{n,m}$, where n is the number of radial nodes and m is the orbital angular momentum. The first five bound state energies found using the outlined approach are compared with an FEM calculation for the spherical potential well with a radius $a = 10$ Å and are included in Tab. 5.5. The generated mesh used in the calculations consisted of 2095 nodes and 12309 elements.

Table 5.5: The first five bound state energy eigenvalues of the finite spherical potential well with $a = 10$Å and $V_0 = 5$ eV using the FEM program with linear and cubic basis functions. The approximation was carried out using 12309 elements.

$E_{n,m}$	Energy eigenvalues (eV) linear basis	Energy eigenvalues (eV) cubic basis	Exact eigenvalues (eV)
(0,0)	0.349	0.317	0.317
(1,1)	0.772	0.651	0.649
(2,2)	1.392	1.071	1.065
(1,0)	1.734	1.258	1.262
(3,3)	2.172	1.578	1.563

Next we benchmark the FEM program against a spherically symmetric quantum dot with a confining potential arising from the conduction band offset surrounding the material. This problem is very similar to the previous spherically symmetric potential well with the conduction band offset causing a finite spherical potential well. However, we use the effective mass approximation in Schödinger's equation with the effective mass varying as follows:

$$m(r)^* = \left\{ \begin{array}{ll} m_D, & 0 \leq r < a \\ m_B, & r > a \end{array} \right\} \tag{5.17}$$

Levels appearing in the conduction band offset are found by factoring the wavefunction into a radial and angular part and solving the radial equation exactly the same way as the spherical potential well except solutions are determined by enforcing continuity at $r = a$:

$$R_{\text{inside}}(r)|_{r=a} = R_{\text{outside}}(r)|_{r=a} \quad \text{and} \quad \frac{1}{m_D}\frac{R_{\text{inside}}(r)}{dr}|_{r=a} = \frac{1}{m_B}\frac{R_{\text{outside}}(r)}{dr}|_{r=a} \tag{5.18}$$

Jenks solved this problem for the quantum dot/barrier pair of $InP_{0.35}Sb_{0.65}/AlAs_{0.17}Sb_{0.83}$ using the approach outlined above for energy levels in the conduction band offset [60]. The bound state energies are compared with our FEM calculation for the parameters included in Tab. 5.6. The generated mesh used in the calculations consisted of 4150 nodes and 24293 elements.

Table 5.6: Conduction band energy levels of a $InP_{0.35}Sb_{0.65}/AlAs_{0.17}Sb_{0.83}$ spherical quantum dot. The parameters $m_D = 0.009 \cdot m_e$, $m_B = 0.131 \cdot m_e$, $a = 35$Å , and $V_0 = 2.15$ eV are used with linear and cubic basis functions to compare against benchmark solution.

$E_{n,m}$	Energy eigenvalues (eV) linear basis	Energy eigenvalues (eV) cubic basis	Exact eigenvalues (eV)
(0,0)	0.610	0.588	0.575
(1,1)	1.901	1.878	1.883

5.6 Remarks

In this chapter, we have discussed in detail all the necessary components of a computer program designed to perform FEM analysis. More specifically, FEM analysis was focused on solving for bound states and related wavefunctions of heterostuctures with a confining potential in two and three dimensions. The FEM program was broken down into three main stages of source code that included: the preprocessing, processing, and post processing stages. The preprocessing stage, 5.2, defines the physical domain of the problem and involves tessellating the region into elements and nodes resulting into a mesh. The processing stage, 5.3, transforms the variational form of Schödinger's equation into matrix operations by evaluating integrals and populating matrices that formulate the generalized eigenvalue problem. The post processing stage, 5.4, solves and visualizes solutions of the generalized eigenvalue problem.

The FEM program source code used to solve for bound states and wavefunctions was thoroughly examined. The main preprocessing source code *QWire_Mesh* and *QDot_Mesh*, used in conjunction with distance functions defining the geometry of the quantum and barrier material, defines a constrained mesh that represents the heterostructure. The main processing source code *FEM_2D_QW* and *FEM_3D_QD* performs the FEM analysis with an option to use linear or cubic basis functions and populates the matrices to form the generalized eigenvalue problem. The main post processing source code *Energy_psi* solves and sorts the solutions to the generalized eigenvalue problem.

Emphasis on flexibility and efficiency was a primary topic that resulted FEM source code design considerations. We discussed the ideas of using linear basis functions to enforce wavefunction continuity at each node and cubic basis functions to enforce wavefunction and derivative continuity at each node. Linear functions result in less accurate solutions than cubic basis functions given the same mesh. However, linear functions require less time to compute than cubic functions do. The flexibility to use either linear or cubic basis functions is a valuable feature that has been built into the source code. It was further discussed that FEM could be more computationally efficient if the integrals were analytically solved and the results were hard coded into the source code. This efficiency measure was exploited and has been built into the source code.

Chapter 6: Strain Induced Potential in Quantum Dot Structures

In the previous two chapters, we developed the finite element method in the context of numerically solving Schrödinger's equation for the purposes of finding QD-IBSC materials. This is necessary as we relax certain restrictive assumptions that are ideal rather than realistic, i.e. geometry of the QD. The finite element program, as described in Chap. 5, was developed with the idea of flexibility not only in terms of accuracy but also in terms of accommodating complicated QD geometries and additional potentials arising in the Hamiltonian other than the offset resulting from the mismatch in materials.

In Chap. 3, we described how self-assembled QDs are realized via S&K growth. The driving force for the growth is a lattice mismatch between barrier and QD material, which induces strain. Remarkably, this method produces uniform coherent QDs that are a necessary requirement for the QD-IBSC. The strain distribution, not surprisingly but largely ignored in the previous chapters, has profound effects on the electronic structure of the QD and must be incorporated into realistic calculations. Strain induces a *strain potential*, V_s, and finds its way in the Hamiltonian as an additional potential. Deformation potential theory, originally described by Bardeen and Shockley [61] and later generalized by Herring and Vogt [62], is used to calculate this strain potential that shifts energy bands resulting from strain in the crystal lattice.

In this chapter, we will investigate how the strain and resulting strain potential are calculated in the framework of the finite element method by invoking the principal of stationary action in the form of minimizing the total strain energy. We begin in Sec. 6.1 by introducing the stress-strain equations, total strain energy, and finite element method used to solve for strain in and around the quantum dot. In Sec. 6.2 we extend the strain discussion to deformation potential theory with the intention of translating the strain into the strain potential. In Sec. 6.3, we briefly present an example of a finite element program written in MATLAB that calculates the stain and resulting strain potential. The output from this program is used as input with the main finite element program of Chap. 5 to account for the strain potential. We conclude in Sec. 6.4.

6.1 Strain-Stress-Displacement Relations for Quantum Dots

The growth of self-assembled QDs in the S&K mode is made possible by an initial strain caused by lattice mismatch between two materials. Take QDs of InAs on barrier material of GaAs, both of which are of the zinc blende structure and characterized by a single lattice constant a_{LC}. The lattice constants differ by about 7.2%, which causes a large stress to develop within the system. The stress is party relieved by the formulation of InAs pyramidal dots. However, a large stress remains and so it is essential to study the influence on the properties of the system.

Several methods can be used to study strain in a QD system but typically fall into two types of approaches: continuum elasticity (CE) and atomistic elasticity (AE). In CE, the system is described as an indefinitely divisible material that neglects all atomic-level information [63, 64]. The system is considered linear and Hooke's law governs. On the other hand, as the name suggests, AE models the atomic displacement field of the crystal with material specific inter-atomic potentials to determine the crystal configuration. The two approaches asymptotically converge with increasing feature size. Both approaches are well suited and justified within their limits. Just as quantum mechanics would not be applied to describing the motion of a cannonball shot from a cannon, AE would not be applied to describing the strain field within an I-beam and vise versa. However, the two methods should be critically evaluated for the application at hand. Boxberg *et al.* has done just this for self-assembled QDs and concluded that CE is sufficient for describing strain within the QD system [65]. Computationally efficient and appropriate for the finite element method, we use CE in our strain calculations throughout the rest of this thesis.

In CE, we view both the QD and barrier material as linearly elastic meaning that each material will return to its original form after being deformed by a force. The relationship between the strain[1], ϵ, and stress[2],σ, is described by Hooke's law. Hooke's law in its simplest form is given by

$$\sigma_x = E\epsilon_x \tag{6.1}$$

$$\epsilon_x = \frac{du}{dx} \tag{6.2}$$

where σ_x is in the x direction, E = modulus of elasticity of the material, and u is the displacement in the x direction. As way of background, we will provide a brief extension of Hooke's law in three dimensions for isotropic material (the amount of material on this subject is extensive and an attempt at categorizing a select few as 'standard' would slight those not included. However references that we found useful on this subject include [66, 67, 68]). As a prerequisite, we must classify the stress and strain as 'normal' or 'shear'. Normal stress and strain act in the direction perpendicular to the surface while shear stress and strain act in the direction along surface. First, consider only normal stress and strain. Normal stress in the x direction produces positive strain in the x direction and we employ Eq. 6.2. However, normal stress in both the y and z (σ_y and σ_z) directions produce negative strains in the x direction via modified version of Eq. 6.2 using the material's Poisson's ratio[3], ν , such that

$$\epsilon_x^{\perp} = -\nu\frac{\sigma_y}{E} - \nu\frac{\sigma_z}{E} \tag{6.3}$$

where $\epsilon_x^{\perp}$ is the strain contribution from normal stresses σ_y and σ_z. Imposing superposition, we arrive at normal strain in the x direction, ϵ_x

$$\epsilon_x = \epsilon_x^{\parallel} + \epsilon_x^{\perp} \tag{6.4}$$

$$\epsilon_x = \frac{\sigma_x}{E} - \nu\frac{\sigma_y}{E} - \nu\frac{\sigma_z}{E} \tag{6.5}$$

Strains in the x and y directions can be determined in a similar manner as in Eq. 6.5 such that

$$\epsilon_y = -\nu\frac{\sigma_x}{E} + \frac{\sigma_y}{E} - \nu\frac{\sigma_z}{E} \tag{6.6}$$

$$\epsilon_z = -\nu\frac{\sigma_x}{E} - \nu\frac{\sigma_y}{E} + \frac{\sigma_z}{E} \tag{6.7}$$

It is convenient to express the normal strains in terms of normal stresses and vice versa in matrix form.

$$\begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \end{bmatrix} = \frac{1}{E}\begin{bmatrix} 1 & -\nu & -\nu \\ -\nu & 1 & -\nu \\ -\nu & -\nu & 1 \end{bmatrix} \cdot \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \end{bmatrix} \tag{6.8}$$

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \end{bmatrix} = E\begin{bmatrix} 1 & -\nu & -\nu \\ -\nu & 1 & -\nu \\ -\nu & -\nu & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \end{bmatrix} \Rightarrow$$

$$\frac{E}{(2\nu-1)(\nu+1)}\begin{bmatrix} \nu-1 & -\nu & -\nu \\ -\nu & \nu-1 & -\nu \\ -\nu & -\nu & \nu-1 \end{bmatrix} \cdot \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \end{bmatrix} \tag{6.9}$$

Equation 6.9 is in a form that we will be able to exploit.

[1] Strain is a description of the deformation in terms of relative displacement.

[2] Stress is a measure of the internal forces acting within a deformable body. Quantitatively, it is a measure of the average force per unit area of a surface within the body on which internal forces act.

[3] Poisson's ratio is the ratio, when a sample object is stretched, of the contraction or transverse strain (perpendicular to the applied load), to the extension or axial strain (in the direction of the applied load).

Hooke's law for normal stress and strain also applies to shear stress, τ , and strain, γ

$$\tau = G\gamma \tag{6.10}$$

where G is the shear modulus analogous to the modulus of elasticity. In three dimensions, there are a total of 3 independent shear strains and stresses such that

$$\tau_{xy} = G\gamma_{xy} \quad \tau_{yz} = G\gamma_{yz} \quad \tau_{zx} = G\gamma_{zx} \tag{6.11}$$

and we can compactly express the normal and shear stresses in a 6×1 vector $\boldsymbol{\sigma}$ in terms of a *stress/strain* 6×6 matrix $\mathbf{C}$ and 6×1 strain vector $\boldsymbol{\epsilon}$

$$\boldsymbol{\sigma} = \boldsymbol{C} \cdot \boldsymbol{\epsilon} \tag{6.12}$$

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{bmatrix} = \frac{E}{(2\nu-1)(\nu+1)} \times \begin{bmatrix} \nu-1 & -\nu & -\nu & 0 & 0 & 0 \\ -\nu & \nu-1 & -\nu & 0 & 0 & 0 \\ -\nu & -\nu & \nu-1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{G(2\nu-1)(\nu+1)}{E} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{G(2\nu-1)(\nu+1)}{E} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{G(2\nu-1)(\nu+1)}{E} \end{bmatrix} \cdot \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix} \tag{6.13}$$

We notice that the stress/strain matrix $\mathbf{C}$ in Eq. 6.13 is symmetric and has three independent elements: C_{11}, C_{12}, and C_{44}. These three independent elastic constants are either experimentally measured or theoretically calculated and presented in the literature, especially for the technologically important compound semiconductors.

Our derivation of Hooke's law in three dimensions, albeit brief and certainly not exhaustive, was done with the idea of applying FEM to calculate the induced strain (and eventually strain potential) caused from the initial strain provided by the lattice mismatch between the QD and barrier materials. To this end, it is incomplete and we must include initial strain effects. To understand how to modify Hooke's law to incorporate initial strains, we look to another physical phenomenon that equivalently acts as an initial strain and is well understood.

A thermal stress results when a body is constrained from deforming as it is subjected to a temperature change and, due to the constraints, causes an initial strain. The system will respond to the initial strain and relax in order to lower its elastic energy. Hooke's law is modified to incorporate this initial strain as follows [67]

$$\boldsymbol{\sigma} = \boldsymbol{C} \cdot (\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_T) \tag{6.14}$$

where $\boldsymbol{\epsilon}$ is the relaxation strain vector and $\boldsymbol{\epsilon}_T$ is the thermal strain vector. The thermal strain is calculated from a material's coefficient of thermal expansion, α, multiplied by temperature change ΔT

$$\boldsymbol{\epsilon}_T = \alpha\, \Delta T \tag{6.15}$$

where α is typically in units of (in./in.)/°F or (mm/mm)/°C. The relaxation strain is then calculated by invoking the principal of stationary action in the form of minimizing the strain energy.

Analogous to the initial strain caused by lattice mismatch between two materials, we adopt the modification to Hooke's law in Eq. 6.14 from thermal stress and will assign a fictitious thermal strain to the QD system. Then, by minimizing the strain energy of the system, U, we find the relaxation strain, which in turn is used to find the strain potential.

We now turn our focus and apply FEM to calculate the strain of the QD system. For the ensuing discussion, we will begin with FEM generalities as we did in Chap. 4 followed by the

specifics, i.e. populating the fictitious thermal strain vector. We will only consider three dimensions and assume tetrahedral elements with four nodes, although the discussion is applicable to one and two dimensions.

Applying FEM in this context begins at defining the strain energy of the system [67]

$$U = \int_\Omega \left(\int_0^{\boldsymbol{\epsilon}} \boldsymbol{\sigma} d\boldsymbol{\epsilon} \right) d\Omega \tag{6.16}$$

By substituting $\boldsymbol{\sigma}$ with $\boldsymbol{C} \cdot (\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_T)$ from Eq. 6.14 and performing the integration with respect to $\boldsymbol{\epsilon}$, we have in matrix form

$$U = \frac{1}{2} \int_\Omega \boldsymbol{\epsilon}^t \cdot \mathbf{C}^t \cdot \boldsymbol{\epsilon} \; d\Omega - \int_\Omega \boldsymbol{\epsilon}^t \cdot \mathbf{C}^t \cdot \boldsymbol{\epsilon}_T \; d\Omega \tag{6.17}$$

$$U = \frac{1}{2} \int_\Omega \boldsymbol{\epsilon}^t \cdot \mathbf{C} \cdot \boldsymbol{\epsilon} \; d\Omega - \int_\Omega \boldsymbol{\epsilon}^t \cdot \mathbf{C} \cdot \boldsymbol{\epsilon}_T \; d\Omega \tag{6.18}$$

where $\boldsymbol{\epsilon}^t$ is the transpose of $\boldsymbol{\epsilon}$ and have made use of $\boldsymbol{C}^t = \boldsymbol{C}$.

The next step is to define $\boldsymbol{\epsilon}$ in terms of unknown nodal displacements, which we will eventually solve for and use to determine the strain. The unknown nodal displacements are given in vector $\boldsymbol{d}$ and relate to $\boldsymbol{\epsilon}$ through the matrix $\boldsymbol{B}$

$$\boldsymbol{\epsilon} = \boldsymbol{B} \cdot \boldsymbol{d} \tag{6.19}$$

Both $\boldsymbol{B}$ and $\boldsymbol{d}$ will be more fully defined but for the purposes of minimizing Eq. 6.18, we make use of Eq. 6.19 to obtain the strain energy in terms of $\boldsymbol{d}$.

$$U = \frac{1}{2} \int_\Omega \boldsymbol{d}^t \cdot \boldsymbol{B}^t \cdot \mathbf{C} \cdot \boldsymbol{B} \cdot \boldsymbol{d} \; d\Omega - \int_\Omega \boldsymbol{d}^t \cdot \boldsymbol{B}^t \cdot \mathbf{C} \cdot \boldsymbol{\epsilon}_T \; d\Omega \tag{6.20}$$

In a similar approach to deriving the integral form of Schödinger's equation (see Eq. 4.1), a variation is placed on the function U such that any variation with respect to the nodal variables in $\boldsymbol{d}$ is stationary. The result is the following

$$\frac{\delta U}{\delta \boldsymbol{d}} = \int_\Omega \boldsymbol{B}^t \cdot \mathbf{C} \cdot \boldsymbol{B} \cdot \boldsymbol{d} \; d\Omega - \int_\Omega \boldsymbol{B}^t \cdot \mathbf{C} \cdot \boldsymbol{\epsilon}_T \; d\Omega = 0 \tag{6.21}$$

$$\int_\Omega \boldsymbol{B}^t \cdot \mathbf{C} \cdot \boldsymbol{B} \cdot \boldsymbol{d} \; d\Omega = \int_\Omega \boldsymbol{B}^t \cdot \mathbf{C} \cdot \boldsymbol{\epsilon}_T \; d\Omega \tag{6.22}$$

The left hand side of the equation, not including $\boldsymbol{d}$, is a matrix called the *stiffness matrix*, while the right side of the equation is a vector called the *force vector*. The vector $\boldsymbol{d}$ contains unknown coefficients that represent the displacement in each spatial direction at a particular node. Therefore, in three dimensions there are three degrees of freedom per node for a total of 12 degrees of freedom for a tetrahedron element. If the number of elements in a mesh is N_E, then the stiffness matrix in Eq. 6.22 turns into

$$\sum_{\alpha=1}^{N_E} \underbrace{\left(\int_{\Omega_\alpha} \boldsymbol{B}^t \cdot \mathbf{C} \cdot \boldsymbol{B} \; d\Omega_\alpha \right)}_{M_{ij}^\alpha} \boldsymbol{d}_j^\alpha \tag{6.23}$$

where M_{ij}^α is the elemental stiffness matrix and $\boldsymbol{d}_j^\alpha$ contains the unknown displacement coefficients within the element. The force vector in Eq. 6.22 turns into

$$\sum_{\alpha=1}^{N_E} \underbrace{\left(\int_{\Omega_\alpha} \boldsymbol{B}^t \cdot \mathbf{C} \cdot \boldsymbol{\epsilon}_T \; d\Omega_\alpha \right)}_{F_i^\alpha} \tag{6.24}$$

where F_i^α is the elemental force vector. Integrals are carried out for each element and placed into

the global algebraic equation

$$M_{ij} d_j = F_i \tag{6.25}$$

Once the boundary conditions are properly treated in Eq. 6.25, we call a matrix inversion to determine the unknown coefficients of vector $\boldsymbol{d}$. The strain in each element is determined from the calculated $\boldsymbol{d}$ and equation Eq. 6.19 (see Sec. 6.3 for specifics).

Analogous to Chap. 4, we represent each elemental displacement function by a linear combination of a finite set of basis functions such that

$$u(x,y,z) = \sum_{i=1}^{n} u_i \phi_i(x,y,z),\ v(x,y,z) = \sum_{i=1}^{n} v_i \phi_i(x,y,z),\ w(x,y,z) = \sum_{i=1}^{n} w_i \phi_i(x,y,z) \tag{6.26}$$

where u, v, and w are the elemental displacement functions in the x, y, and z directions respectively, u_i, v_i, and w_i are the unknown coefficients of vector $\boldsymbol{d}$, $\phi_i(x,y,z)$ are basis functions, and the summation's upper limit is the number of basis functions per element. For our application, we choose basis functions that have a linear polynomial form (see Eq. 4.56). Again, we define a benchmark element with the intention of linearly mapping all other elements in the mesh to the benchmark element. We assume the benchmark element is the standard tetrahedron and is defined in local coordinates (ξ,η,ζ) with vertices located at $(0,0,0)$, $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$. Again, the basis functions are defined in Eq. 4.58 and repeated below for convenience.

$$\begin{aligned} \phi_1(\xi,\eta,\zeta) &= 1-\xi-\eta-\zeta \\ \phi_2(\xi,\eta,\zeta) &= \xi \\ \phi_3(\xi,\eta,\zeta) &= \eta \\ \phi_4(\xi,\eta,\zeta) &= \zeta \end{aligned} \tag{6.27}$$

The coordinate transformation between the $(\xi,\eta,\zeta) \rightarrow (x,y,z)$ can be expressed as a linear combination of the basis functions (see Eq. 4.59). To define the elemental stiffness matrix and elemental force vector, we consider the first element in a mesh and assume it contains nodes $n = 1,2,3,4$ for the tetrahedron. In matrix form, the expressions for u, v, and w in first element are

$$\begin{bmatrix} u(x,y,z) \\ v(x,y,z) \\ w(x,y,z) \end{bmatrix} \rightarrow \begin{bmatrix} \phi_1 & 0 & 0 & \phi_2 & 0 & 0 & \phi_3 & 0 & 0 & \phi_4 & 0 & 0 \\ 0 & \phi_1 & 0 & 0 & \phi_2 & 0 & 0 & \phi_3 & 0 & 0 & \phi_4 & 0 \\ 0 & 0 & \phi_1 & 0 & 0 & \phi_2 & 0 & 0 & \phi_3 & 0 & 0 & \phi_4 \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ v_1 \\ w_1 \\ \cdot \\ \cdot \\ \cdot \\ u_4 \\ v_4 \\ w_4 \end{bmatrix} \tag{6.28}$$

6.1.1 Elemental Stiffness Matrix

We begin by examining the stiffness matrix in three dimensions. It is constructed from Eq. 6.23

$$\boldsymbol{M} = \int\int\int \boldsymbol{B}^t \cdot \mathbf{C} \cdot \boldsymbol{B}\ dx\,dy\,dz \tag{6.29}$$

where $\boldsymbol{C}$ is the stress/strain matrix as defined in Eq. 6.13 above but the matrix $\boldsymbol{B}$ is still unknown. It is well known that strain in three dimensions is related to the displacement as follows [67, 69]

$$\begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix} = \begin{bmatrix} \frac{\partial u(x,y,z)}{\partial x} \\ \frac{\partial v(x,y,z)}{\partial y} \\ \frac{\partial w(x,y,z)}{\partial z} \\ \frac{\partial u(x,y,z)}{\partial y} + \frac{\partial v(x,y,z)}{\partial x} \\ \frac{\partial v(x,y,z)}{\partial z} + \frac{\partial w(x,y,z)}{\partial y} \\ \frac{\partial w(x,y,z)}{\partial x} + \frac{\partial u(x,y,z)}{\partial z} \end{bmatrix} \tag{6.30}$$

We can utilize the elements of the 3×3 inverse Jacobian, $\mathbf{J}^{-1}$, for the necessary coordinate transformation. The column vector containing the partial derivatives of the elemental displacement functions with respect to the global coordinates as in Eq. 6.30 is represented in the following form

$$\begin{bmatrix} \frac{\partial u(x,y,z)}{\partial x} \\ \frac{\partial v(x,y,z)}{\partial y} \\ \frac{\partial w(x,y,z)}{\partial z} \\ \frac{\partial u(x,y,z)}{\partial y} + \frac{\partial v(x,y,z)}{\partial x} \\ \frac{\partial v(x,y,z)}{\partial z} + \frac{\partial w(x,y,z)}{\partial y} \\ \frac{\partial w(x,y,z)}{\partial x} + \frac{\partial u(x,y,z)}{\partial z} \end{bmatrix} =$$

$$\begin{bmatrix} J_{11}^{-1} & 0 & 0 & J_{21}^{-1} & 0 & 0 & J_{31}^{-1} & 0 & 0 \\ 0 & J_{12}^{-1} & 0 & 0 & J_{22}^{-1} & 0 & 0 & J_{32}^{-1} & 0 \\ 0 & 0 & J_{13}^{-1} & 0 & 0 & J_{23}^{-1} & 0 & 0 & J_{33}^{-1} \\ J_{12}^{-1} & J_{11}^{-1} & 0 & J_{22}^{-1} & J_{21}^{-1} & 0 & J_{32}^{-1} & J_{31}^{-1} & 0 \\ 0 & J_{13}^{-1} & J_{12}^{-1} & 0 & J_{23}^{-1} & J_{22}^{-1} & 0 & J_{33}^{-1} & J_{32}^{-1} \\ J_{13}^{-1} & 0 & J_{11}^{-1} & J_{23}^{-1} & 0 & J_{21}^{-1} & J_{33}^{-1} & 0 & J_{31}^{-1} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial u(x,y,z)}{\partial \xi} \\ \frac{\partial v(x,y,z)}{\partial \xi} \\ \frac{\partial w(x,y,z)}{\partial \xi} \\ \frac{\partial u(x,y,z)}{\partial \eta} \\ \frac{\partial v(x,y,z)}{\partial \eta} \\ \frac{\partial w(x,y,z)}{\partial \eta} \\ \frac{\partial u(x,y,z)}{\partial \zeta} \\ \frac{\partial v(x,y,z)}{\partial \zeta} \\ \frac{\partial w(x,y,z)}{\partial \zeta} \end{bmatrix} \tag{6.31}$$

where $(J^{-1})_{ij}$ refers to the individual elements of $\mathbf{J}^{-1}$. The 6×9 matrix on the right hand side of Eq. 6.31 we call $\boldsymbol{\jmath}$. The column vector on the right hand side containing the partial derivatives of the elemental displacement functions is represented by

$$\begin{bmatrix} \phi_{1,\xi} & 0 & 0 & \phi_{2,\xi} & 0 & 0 & \phi_{3,\xi} & 0 & 0 & \phi_{4,\xi} & 0 & 0 \\ 0 & \phi_{1,\xi} & 0 & 0 & \phi_{2,\xi} & 0 & 0 & \phi_{3,\xi} & 0 & 0 & \phi_{4,\xi} & 0 \\ 0 & 0 & \phi_{1,\xi} & 0 & 0 & \phi_{2,\xi} & 0 & 0 & \phi_{3,\xi} & 0 & 0 & \phi_{4,\xi} \\ \phi_{1,\eta} & 0 & 0 & \phi_{2,\eta} & 0 & 0 & \phi_{3,\eta} & 0 & 0 & \phi_{4,\eta} & 0 & 0 \\ 0 & \phi_{1,\eta} & 0 & 0 & \phi_{2,\eta} & 0 & 0 & \phi_{3,\eta} & 0 & 0 & \phi_{4,\eta} & 0 \\ 0 & 0 & \phi_{1,\eta} & 0 & 0 & \phi_{2,\eta} & 0 & 0 & \phi_{3,\eta} & 0 & 0 & \phi_{4,\eta} \\ \phi_{1,\zeta} & 0 & 0 & \phi_{2,\zeta} & 0 & 0 & \phi_{3,\zeta} & 0 & 0 & \phi_{4,\zeta} & 0 & 0 \\ 0 & \phi_{1,\zeta} & 0 & 0 & \phi_{2,\zeta} & 0 & 0 & \phi_{3,\zeta} & 0 & 0 & \phi_{4,\zeta} & 0 \\ 0 & 0 & \phi_{1,\zeta} & 0 & 0 & \phi_{2,\zeta} & 0 & 0 & \phi_{3,\zeta} & 0 & 0 & \phi_{4,\zeta} \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ v_1 \\ w_1 \\ \cdot \\ \cdot \\ \cdot \\ u_4 \\ v_4 \\ w_4 \end{bmatrix} \tag{6.32}$$

where the comma after the subscript indicates differentiation with respect to the variable that follows. We have made use of the benchmark elemental displacement function from Eq. 6.28 and decomposed it into a 9×12 matrix $\boldsymbol{B'}$. The abbreviated column vector in Eq. 6.32 is just the unknown coefficients of vector $\boldsymbol{d}$ for which we are solving for. Using Eq. 6.31 and Eq. 6.32, we obtain the 6×12 matrix $\boldsymbol{B}$

$$\boldsymbol{B} = \boldsymbol{\jmath} \cdot \boldsymbol{B'} \tag{6.33}$$

Transforming the integral of Eq. 6.29 into local coordinates, which requires the Jacobian determinant $|J|$, will represent the elemental stiffness matrix:

$$\int_0^1 \int_0^{1-\eta} \int_0^{1-\eta-\zeta} \boldsymbol{B'}^t \cdot \boldsymbol{\jmath}^t \cdot \mathbf{C} \cdot \boldsymbol{\jmath} \cdot \boldsymbol{B'} \, |J| \; d\xi \, d\eta \, d\zeta \tag{6.34}$$

with the limits of integration over the standard tetrahedron. One of the advantages of defining the benchmark element is to carry out the integrals once so that the extra computational operations required to perform numerical integration techniques are not employed. In Eq. 6.34, we observe that the entries of each matrix are constant as well as the Jacobian determinant and can be brought outside the integral. For the problem at hand, the integrals are carried out and the elemental stiffness matrix is transformed into the following

$$\boldsymbol{M} = |J| \times \frac{1}{6} \cdot \boldsymbol{B'}^t \cdot \boldsymbol{\jmath}^t \cdot \mathbf{C} \cdot \boldsymbol{\jmath} \cdot \boldsymbol{B'} \tag{6.35}$$

The elemental stiffness matrix is a 12×12 matrix. Now it only remains to assemble each elemental matrix entry in the global $3N \times 3N$ stiffness matrix for an N node mesh.

To the best of our knowledge, the approach that we have taken in determining $\boldsymbol{B}$ above is original. Most techniques tend to calculate the coefficients of the basis functions for every element in the mesh and directly determine $\boldsymbol{B}$ rather than use the basis functions of Eq. 6.28. To verify our approach, we consider a tetrahedral element with its four nodes located $(1,1,2)$, $(0,0,0)$, $(0,2,0)$, and $(2,1,0)$. The matrix $\boldsymbol{B}$ has been determined (see Example 12.1 in reference [67]) as

$$\boldsymbol{B} = \begin{bmatrix} 0 & 0 & 0 & -\frac{1}{4} & 0 & 0 & -\frac{1}{4} & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & -\frac{1}{8} & 0 & 0 & -\frac{1}{8} & 0 & 0 & -\frac{1}{4} \\ 0 & 0 & 0 & -\frac{1}{2} & -\frac{1}{4} & 0 & \frac{1}{2} & -\frac{1}{4} & 0 & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & 0 & -\frac{1}{8} & -\frac{1}{2} & 0 & -\frac{1}{8} & \frac{1}{2} & 0 & -\frac{1}{4} & 0 \\ \frac{1}{2} & 0 & 0 & -\frac{1}{8} & 0 & -\frac{1}{4} & -\frac{1}{8} & 0 & -\frac{1}{4} & -\frac{1}{4} & 0 & \frac{1}{2} \end{bmatrix} \tag{6.36}$$

Using Eq. 6.32, $\boldsymbol{B'}$ is determined from the basis functions

$$\boldsymbol{B'} = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.37}$$

while $\boldsymbol{\jmath}$ is determined by evaluating the Jacobian, taking its inverse, and then placing the inverse Jacobian's elements in Eq. 6.31. The Jacobian matrix is evaluated from the nodal coordinates given in the example and its inverse is taken as

$$\boldsymbol{J} = \begin{bmatrix} -1 & -1 & 1 \\ -1 & 1 & 0 \\ -2 & -2 & -2 \end{bmatrix} \qquad \boldsymbol{J}^{-1} = 8 \cdot \begin{bmatrix} -2 & -4 & -1 \\ -2 & 4 & -1 \\ 4 & 0 & -2 \end{bmatrix} \tag{6.38}$$

Substituting the results from the inverse Jacobian into Eq. 6.31, we obtain $\boldsymbol{\jmath}$. Finally from Eq. 6.33, $\boldsymbol{B}$ is evaluated. The resulting 6×12 matrix, being arduous to evaluate longhand is best left to the computer, is the same as Eq. 6.36.

6.1.2 Elemental Force Vector

We begin by examining the force vector in three dimensions. It is constructed from Eq. 6.24

$$\boldsymbol{F} = \int\int\int \boldsymbol{B}^t \cdot \mathbf{C} \cdot \boldsymbol{\epsilon}_T \; dx\, dy\, dz \tag{6.39}$$

where $\boldsymbol{C}$ is the 12×12 stress/strain matrix as defined in Eq. 6.13 above, $\boldsymbol{B}$ is the 6×12 matrix defined in the previous section, and $\boldsymbol{\epsilon}_T$ is a fictitious 6×1 thermal strain vector representing an initial strain. The initial strain is taken as the difference between the lattice constants of QD and the barrier with respect to the barrier such that the initial strain ϵ_0 is [70]

$$\epsilon_0 = \frac{a_{LC}^{QD} - a_{LC}^{B}}{a_{LC}^{B}} \tag{6.40}$$

where a_{LC}^{QD} and a_{LC}^{B} are the lattice constants of the QD and barrier material respectively. The initial strain is measured with respect to the barrier material and is modeled such that it only occurs within the dot [71, 72, 73]. Thus, elements within the barrier material do not experience an initial strain. In addition, during self-assembled QD growth, the in-plane direction experiences this initial strain only as a dilation, meaning shear components are zero [74]. It is generally assumed that the in-plane direction initial strain is accompanied by a purely dilation out-of-plane initial strain proportional to ϵ_0 [65, 75]. The initial out-of-plain strain is found using Eq. 6.13, where we solve for

$$\begin{aligned} 0 &= C_{12}(\epsilon_0 + \epsilon_0) + C_{11}\epsilon_{zz} \\ \epsilon_{zz} &= -\frac{2C_{12}}{C_{11}}\epsilon_0 \end{aligned} \tag{6.41}$$

In light of this information and in the context of Eq. 6.15, the fictitious thermal strain vector is just

$$\begin{aligned} \boldsymbol{\epsilon}_T &= \left\{ \begin{array}{ll} \Delta T \cdot [0\ 0\ 0\ 0\ 0\ 0]^t, & \text{Barrier} \\ \Delta T \cdot \left[\epsilon_0\ \epsilon_0\ -\frac{2C_{12}}{C_{11}}\epsilon_0\ 0\ 0\ 0\right]^t, & \text{QD} \end{array} \right\} \\ \Delta T &= 1 \end{aligned} \tag{6.42}$$

where ϵ_0 is just Eq. 6.40 and the fictitious rise in temperature is $\Delta T = 1$.

Transforming the integral of Eq. 6.39 into local coordinates, which requires the Jacobian determinant $|J|$, will represent the elemental force vector:

$$\int_0^1 \int_0^{1-\eta} \int_0^{1-\eta-\zeta} \boldsymbol{B'}^t \cdot \boldsymbol{\jmath}^t \cdot \mathbf{C} \cdot \boldsymbol{\epsilon}_T \, |J| \; d\xi\, d\eta\, d\zeta \tag{6.43}$$

with the limits of integration over the standard tetrahedron. We have made use of Eq. 6.33, as we did in the elemental stiffness matrix construction. All the entries making up Eq. 6.43 are constant as well as the Jacobian determinant and can be brought outside the integral. The integrals are carried out and the elemental stiffness matrix is transformed into the following

$$\boldsymbol{M} = |J| \times \frac{1}{6} \cdot \boldsymbol{B'}^t \cdot \boldsymbol{\jmath}^t \cdot \mathbf{C} \cdot \boldsymbol{\epsilon}_T \tag{6.44}$$

The elemental stiffness matrix is a 12×1 vector. Now it only remains to assemble each elemental matrix entry in the global $3N \times 1$ force vector for an N node mesh.

6.1.3 Boundary Conditions

Following the prescription above, it is possible to construct the global algebraic equation Eq. 6.25 necessary to determine the unknown nodal displacements and eventually the strain in each element. However before a matrix inversion is called, the matrix $\boldsymbol{M}$ and vector $\boldsymbol{F}$ must be treated for boundary

conditions. Understanding the problem is paramount to understand how we treat the rows and columns of the stiffness matrix and rows of the force vector that represents the nodal displacement values occurring on the bounding box of the mesh. As an example (see benchmarked example in Sec. 6.3), let's consider a QD that is buried in an infinite amount of barrier material. The assumption of an infinite amount of barrier material corresponds to the situation that exists when the QD is deeply buried in the barrier material and the boundaries do not impact the strain state. The displacement at points approaching infinity must decay to zero. For strain calculations using FEM, this type of boundary condition is enforced by requiring the nodal displacements on the mesh's bounding box be zero. This is necessary because, in most cases, the stiffness matrix is singular meaning that applying boundary conditions, artificial or not, is a requirement to coax the matrix inversion needed to calculate displacement and strain. The rows and columns in the stiffness matrix corresponding to those displacement values in all three spacial directions on the mesh boundary are removed. Similarly, those same rows in the force vector are removed. At this point the stiffness matrix can be inverted and subsequently multiplied by the force vector. In this example, the stiffness matrix and force vector are reduced in size by three times the number of nodes on the bounding box. It is important to analyze the solution in order to verify that the artificial boundaries did not impact the results. We suggest a comparison of two differently sized meshes. If differences occur, it is likely that the artificial boundaries had an influence on the solution. The analysis must be repeated until the two results converge.

6.2 Strain Potential

Strained heterostructures induce a non-negligible potential, V_s, that can be determined by deformation potential theory. Bardeen and Shockley originally formulated the theory in the context of non-degenerate energy bands and it was later generalized by Herring and Vogt [61, 62]. Bir and Pikus specifically applied the theory to strained materials [76]. In this section, we introduce the origins of the strain potential and highlight the important aspects.

Let us define the coordinates before and after the deformation as $\boldsymbol{r}$ and $\boldsymbol{r'}$ and the strain vector $\boldsymbol{\epsilon}$. Therefore, we have the relation

$$\boldsymbol{r'} - \boldsymbol{r} = \boldsymbol{\epsilon} \cdot \boldsymbol{r} \tag{6.45}$$

where we have used a slightly different notation for the strain, which is defined in its matrix form [77].

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_x & \gamma_{xy}/2 & \gamma_{xz}/2 \\ \gamma_{xy}/2 & \epsilon_y & \gamma_{yz}/2 \\ \gamma_{xz}/2 & \gamma_{yz}/2 & \epsilon_z \end{bmatrix} = \begin{bmatrix} \epsilon_{xx} & \epsilon_{xy} & \epsilon_{xz} \\ \epsilon_{xy} & \epsilon_{yy} & \epsilon_{yz} \\ \epsilon_{xz} & \epsilon_{yz} & \epsilon_{zz} \end{bmatrix} \tag{6.46}$$

with an inversion, we get

$$\boldsymbol{r} = (1 - \boldsymbol{\epsilon}) \cdot \boldsymbol{r'} \tag{6.47}$$

$$r_i \approx r'_i - \sum_j \epsilon_{ij} r'_j \tag{6.48}$$

The deformation potential originates from transforming Schrödinger's equation from the new coordinates to the old coordinates. The differential operator ∇' expressed in the old coordinates using Eq. 6.48 and the chain rule is

$$\frac{\partial}{\partial r'_i} = \sum_j \frac{\partial r_j}{\partial r'_i} \cdot \frac{\partial}{\partial r_j} = \frac{\partial}{\partial r_i} - \sum_j \epsilon_{ji} \frac{\partial}{\partial r_j} \tag{6.49}$$

$$\nabla' = (1 - \boldsymbol{\epsilon}) \cdot \nabla \tag{6.50}$$

Therefore, $\nabla' \cdot \nabla'$ is just

$$\nabla' \cdot \nabla' = \nabla \cdot \nabla - 2 \sum_{i,j} \nabla_i \cdot \epsilon_{ij} \cdot \nabla_j \tag{6.51}$$

where the higher terms of ϵ have been neglected. Expanding the potential $V(\boldsymbol{r'})$ in terms of the old coordinates is

$$V(\boldsymbol{r'}) = V[(1+\boldsymbol{\epsilon})\cdot\boldsymbol{r}] = V_0(\boldsymbol{r}) + \sum_{i,j} V_{ij}\epsilon_{ij}, \qquad V_{ij} = \frac{\partial V}{\partial \epsilon_{ij}} \mid_{\epsilon_{ij}\to 0} \tag{6.52}$$

As shown above in Eqs. 6.51 and 6.52, Schrödinger's equation includes additional terms corresponding to the strain and can be grouped into a strain potential

$$V_s = \frac{\hbar^2}{m^*}\sum_{ij}\nabla_i\cdot\epsilon_{ij}\cdot\nabla_j + \sum_{ij} V_{ij}\epsilon_{ij} = \sum_{ij} D^{ij}\epsilon_{ij} \tag{6.53}$$
$$D^{ij} = \frac{\hbar^2}{m^*}\nabla_i\cdot\nabla_j + V_{ij}$$

where D is called the deformation potential tensor and has the units of energy. As with the elasticity constants, the deformation potential tensor elements are empirically derived and found in the literature containing band parameters. In the conduction band, the change in the conduction-band-edge energy due to strain can be described by one deformation constant and the hydrostatic strain such that the strain potential is

$$V_s^c = -a_c(\epsilon_{xx}+\epsilon_{yy}+\epsilon_{zz}) \tag{6.54}$$

where the superscript refers to the respective band, in this case the conduction band. In the valence band, a more complex picture emerges. A single deformation constant is insufficient to describe the shift in the valence-band-edge energy. This is due to the degenerate nature of the valence band: interaction between the heavy, light, and spin-orbit split off bands. There are two additional potentials b and d needed to describe the shear terms that split the degeneracy. This is described as the Bir-Pikus strain interaction [76]. The heavy and light band strain potential is given as

$$V_s^{v1} = -a_v(\epsilon_{xx}+\epsilon_{yy}+\epsilon_{zz}) - 3b[(L_x^2 - \frac{1}{3}L^2)\epsilon_{xx} + cp]$$
$$-\sqrt{3}d[(L_xL_y + L_yL_x)\epsilon_{xy} + cp] \tag{6.55}$$

where L is the angular momentum operator and cp is cyclic permutation with respect to the indices x, y, z. The spin-orbit split off potential is given as

$$V_s^{v2} = -a_v(\boldsymbol{L}\cdot\boldsymbol{\sigma})\cdot(\epsilon_{xx}+\epsilon_{yy}+\epsilon_{zz}) - 3b[(L_x\sigma_x - \frac{1}{3}\boldsymbol{L}\cdot\boldsymbol{\sigma})\epsilon_{xx} + cp]$$
$$-\sqrt{3}d[(L_x\sigma_y + L_y\sigma_x)\epsilon_{xy} + cp] \tag{6.56}$$

where $\boldsymbol{\sigma}$ is the Pauli matrix vector.

The strained potentials above are dependent on the strain components as calculated using the procedure in Sec. 6.1 and deformation potential tensor. Due to the breadth of available literature on the experimentally found deformation potential constants, the bulk of work is in calculating the strain. Once the strain components of the strain matrix are found, it becomes trivial to solve for the elemental strain potential.

While the strain potentials arising from deformation potential theory that results in Eqs. 6.54-6.56 are correct, context within the larger discussion of k·p perturbation theory that takes place in Chap. 7 is necessary. By doing so, we will explicitly not only put deformation potential theory in perspective but effective mass theory as well.

6.3 Example of a FEM Strain Program

In this section, we follow the prescription from Sec. 6.1 and offer a MATLAB program that is used to calculate the strain in and around a QD with the intention of translating the strain into elemental strain potentials in such a format that is compatible with the FEM processing code from Sec. 5.3.

We then benchmark the program output to known analytic solutions as we did previously for FEM programs. Appropriately, program logic is similar to the program logic of the FEM processing code.

As with the previous FEM code, instead of approximating all the integrals using quadrature, we have carried out the integrals analytically on a benchmark element and 'hardcoded' the results. The benchmark element is the standard tetrahedron with vertices $(0,0,0)$, $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$. Each element from the mesh is linearly mapped to the benchmarked element to ensure the Jacobian entries and its determinant are constant. The step requiring integral evaluation has been removed and the code's objectives are reduced to: populating the stiffness matrix and force vector, performing a matrix inversion to determine nodal displacements, determining elemental strain from nodal displacements, and finally determining strain potential. The ideas following in the next few paragraphs explain the general structure of the program and highlight the more important aspects.

There are a number of variables that must be set before the program begins. The first variable, **a**, is used to assign the lattice constant of the QD and barrier materials in Angstrom units. The rest of the variables pertain to the quantum and barrier material's elasticity constants and deformation potential constants: elasticity constants expressed in units of stress expressed giga-pascals (GPa), and deformation potential constants expressed in the electron-volt (eV) unit.

Unlike the previous FEM processing code, we have not included the option for the user to choose the type of basis functions. Instead, the linear basis functions from Eq. 6.28 are used to ensure nodal continuity and are constructed from the standard tetrahedron. After the variables are set, the code begins evaluating each element in the mesh. This entails constructing the elemental stiffness matrix and force vector. Each is populated using the benchmarked integral results, various properties of the Jacobian matrix, stress/strain matrix containing the elasticity constants, and, in the force vector's case, the initial strain vector.

After each elemental matrix is constructed, it must be properly overlaid in the larger respective global matrix or vector. A map linking the two is needed. The map or index system must take into consideration how unknown displacements at each node should be organized. If the following notation is used to describe unknown displacements at node i,

$$d_{i,1} = u_1 \quad d_{i,2} = v_1 \quad d_{i,3} = w_1 \tag{6.57}$$

our code wishes to organize these unknowns in the global stiffness matrix and global force vector as follows:

$$\boldsymbol{\psi} = \begin{bmatrix} d_{1,1} \\ d_{1,2} \\ d_{1,3} \\ d_{2,1} \\ d_{2,2} \\ d_{2,3} \\ d_{3,1} \\ d_{3,2} \\ \vdots \end{bmatrix} \tag{6.58}$$

With the desired global organization outlined in Eg. 6.58, the indexing assigns the system degree of freedom to each node using global node numbers and three degrees of freedom per node. The index function, *index_3D*, similar to the function of the same name in the processing code produces the organization of Eq. 6.58 while avoiding inefficient nested 'for loops'.

After global matrices are constructed, the next step is to apply the problem's boundary conditions as discussed in Sec. 6.1.3. This involves post-treating columns and rows of those nodes located on the bounding box. The rows and columns in the stiffness matrix corresponding to nodal displacement values that are to be set to zero on the mesh boundary are removed. Similarly, those same rows in the force vector are removed. The function *dofbc* performs this task. At this point, the stiffness matrix can be inverted and subsequently multiplied by the force vector to obtain the displacement values at each node. Finally, after the displacement values are found, the code runs through all the

elements to determine the strain (and strain potential) using Eq. 6.19. We should note that in order to obtain the total strain in each element, the initial mismatch strain vector is subtracted from the induced relaxation strain.

The source code for the program calculating the strain is contained below with a description of every input/output argument, variable, and function preceded by the %. We will not offer a detailed line by line explanation as done in the previous chapter since the logic and subroutines are similar to the FEM processing code and simple descriptions are included in the code below.

```
function [ d e] = FEM_3D_Strain(p,t,vert)
%FEM_3D_Strain This is finite element code that is designed to find the
%the strain for a self-assembled QD.
%
%
%   FEM_3D_Strain(P,T,VERT,V_NODES)
%       D: DISPLACEMENT VALUES AT EACH NODE (NNODES*3 X 1)
%       E: STRAIN VALUE FOR EACH ELEMENT (NEL X 1)
%       P: COORDINATES OF NODES (NNODES X 3)
%       T: TETRAHEDRAL ELEMENTS (NEL X NNEL)
%       VERT: VERTICES OF QUANTUM DOT (M X P,M VERTICES IN P DIMENSIONS)
%-------------------------------------------------------------------------
%
%   Physical constants:
%       a: lattice constant of QD and barrier material, units of A (1 X 2)
%       C: elastic constants of QD and barrier material, units of GPa (3 X 2)
%       e_i: initial strain
%   Variables:
%       ndof: number of degrees of freedom (dof) per node
%       nnel: number of nodes per element
%       nel: number of elements
%       nnodes: number of nodes
%       enodes: nodes on the bounding box
%       B: derivative matrix (9 X 12)
%       interface_nodes: nodes on interface separating the two materials
%       elem_size: size of stiffness matrix and force vector
%       r,c,v: row & column indices of respective element matrix or vector
%              that point to value v.
%       nd: local node indexing (1 X NNEL)
%       xcoord/ycoord: local coordinate indexing (1 X NNEL)
%       C: elastic properties of element
%       E_o: Initial strain vector (6 X 1)
%       Jac: Jacobian Matrix (3 X 3)
%       J: inverse Jacobian (3 X 3)
%       D: Elastic constant matrix (6 X 6)
%       J_mod: Modified Jacobian matrix from Eq. 6.31 (6 X 9)
%       elem_S: Elemental stiffness matrix (12 X 12)
%       elem_F: Elemental force matrix (12 X 1)
%       index: System degree of freedom assigned to element node(1 X 12)
%       glob_S: Global stiffness matrix
%       glob_F: Global force matrix
%       bc_index: degree of freedom assigned to boundary condition nodes
%       glob_S_int: Global stiffness matrix that has been treated for
%                   boundary conditions
%       glob_F_int: Global force matrix that has been treated for boundary
%                   conditions
%       d_int: nodal displacements not including bc_index displacements
%
%   Functions:
%       surftri: Find surface triangles from mesh
%       inhull: Tests if a set of points is inside a convex hull
```

```
%       jacob_3D: The Jacobian for 3D mapping
%       Elem_matrix_S_3D: Computes the elemental stiffness matrix
%       Elem_matrix_F_3D: Computes the elemental force vector
%       index_3D: Assigns the system dof to the element node
%       assemble: Builds up row & column indices
%       sparse: assembles global matrices by creating a sparse matrix
%       dofbc: returns those nodes on the bounding box that have a boundary
%              condition
%       apply_bc: Applies the boundary conditions to a global matrix or
%                 vector, i.e. removes rows and columns corresponding to
%                 bc_index
%
%ENTER USER INPUT HERE-----------------------------------------------------
%(1) represents dot and (2) represents barrier
a(1)=5.472;
a(2)=5.430;
C_11(1)=1.603;
C_12(1)=0.6188;
C_44(1)=0.4921;
C_11(2)=1.675;
C_12(2)=0.650;
C_44(2)=0.5125;
%END USER INPUT-------------------------------------------------------------
%
e_i=(a(1)-a(2))/a(2);
ndof = 3;
nnel = 4;
nel=length(t(:,1));
nnode=length(p(:,1));
d = zeros(nnode*ndof,1);
B = [-1 0 0 1 0 0 0 0 0 0 0 0; 0 -1 0 0 1 0 0 0 0 0 0 0;...
    0 0 -1 0 0 1 0 0 0 0 0 0; -1 0 0 0 0 0 1 0 0 0 0 0;...
    0 -1 0 0 0 0 0 1 0 0 0 0; 0 0 -1 0 0 0 0 0 1 0 0 0;...
    -1 0 0 0 0 0 0 0 0 1 0 0; 0 -1 0 0 0 0 0 0 0 0 1 0;...
    0 0 -1 0 0 0 0 0 0 0 0 1];
interface_nodes = find(ismember(p,vert,'rows'));
%
enodes = unique(surftri(p,t));
%
e = zeros(nel,6);
S_elem_size = nnel*ndof*nnel*ndof;
F_elem_size = nnel*ndof;
r_S = zeros(nel,S_elem_size);r_F = zeros(nel,F_elem_size);
c_S = zeros(nel,S_elem_size);
v_S = zeros(nel,S_elem_size);v_F = zeros(nel,F_elem_size);
%
%Determine if points are inside, outside, or on QD.  IN is a logical vector
%containing 1's for points inside and on QD surface and 0 for points
%outside.
IN = inhull(p,vert,[],0.01);
%QD_elements is a scalar that sums each element's
%vertices as represented by 1's and 0's.  If this quantity equals nnodes,
%element lies inside QD.
QD_elements = sum(IN(t),2);
%
for iel=1:nel
    nd = t(iel,:);
    xcoord = p(nd,1);
    ycoord = p(nd,2);
```

```
    zcoord = p(nd,3);
    if  (QD_elements(iel) == nnel)
        C(1) = C_11(1);
        C(2) = C_12(1);
        C(3) = C_44(1);
        E_o = [e_i;e_i;e_i;0;0;0];
    else
        C(1) = C_11(2);
        C(2) = C_12(2);
        C(3) = C_44(2);
        E_o = [0;0;0;0;0;0];
    end
    D = [C(1) C(2) C(2) 0 0 0 ; C(2) C(1) C(2) 0 0 0 ;...
        C(2) C(2) C(1) 0 0 0 ; 0 0 0 C(3) 0 0 ; 0 0 0 0 C(3) 0 ;...
        0 0 0 0 0 C(3)];
    Jac = jacob_3D(xcoord,ycoord,zcoord);
    J = inv(Jac);
    J_mod = [J(1,1) 0 0 J(2,1) 0 0 J(3,1) 0 0;...
            0 J(1,2) 0 0 J(2,2) 0 0 J(3,2) 0 ;...
            0 0 J(1,3) 0 0 J(2,3) 0 0 J(3,3) ;...
            J(1,2) J(1,1) 0 J(2,2) J(2,1) 0 J(3,2) J(3,1) 0 ;...
            0 J(1,3) J(1,2) 0 J(2,3) J(2,2) 0 J(3,3) J(3,2) ;...
            J(1,3) 0 J(1,1) J(2,3) 0 J(2,1) J(3,3) 0 J(3,1)];
    %
    %Construct the elemental matrix elem_S by using function Elem_matrix_S
    elem_S = Elem_matrix_S_3D(J_mod,D,B,Jac);
    %Construct the elemental vector elem_F by using function Elem_matrix_F
    elem_F = Elem_vector_F_3D(J_mod,D,B,E_o,Jac);
    %
    %Builds up row & column indices using assemble function
    %
    index = index_3D(nd,ndof);
    [r_S(iel,:) c_S(iel,:) v_S(iel,:)] = assemble(elem_S,index);
    r_F(iel,:) = index;
    v_F(iel,:) = elem_F;

end
%
%Construct the global matrix and vector using sparse and triples of
%rows/columns/values
%
glob_S = sparse(r_S,c_S,v_S,ndof*nnode,ndof*nnode);
glob_F = sparse(r_F,1,v_F,ndof*nnode,1);
%
bc_index = dofbc(enodes,ndof,p,1);
glob_S_int = apply_bc( glob_S,bc_index,1 );
glob_F_int = apply_bc( glob_F,bc_index,2 );
%
d_int = glob_S_int \ glob_F_int;
ind = 1:nnode*ndof;
ind(bc_index)=[];
d(ind)=d_int;
%Determine the strain in each element
for iel=1:nel
    nd = t(iel,:);
    d_dof = dofbc(nd,ndof,[],1);
    xcoord = p(nd,1);
    ycoord = p(nd,2);
    zcoord = p(nd,3);
```

```
    Jac = jacob_3D(xcoord,ycoord,zcoord);
    J = inv(Jac);
    J_mod = [J(1,1) 0 0 J(2,1) 0 0 J(3,1) 0 0;...
            0 J(1,2) 0 0 J(2,2) 0 0 J(3,2) 0 ;...
            0 0 J(1,3) 0 0 J(2,3) 0 0 J(3,3) ;...
            J(1,2) J(1,1) 0 J(2,2) J(2,1) 0 J(3,2) J(3,1) 0 ;...
            0 J(1,3) J(1,2) 0 J(2,3) J(2,2) 0 J(3,3) J(3,2) ;...
            J(1,3) 0 J(1,1) J(2,3) 0 J(2,1) J(3,3) 0 J(3,1)];
    if  (QD_elements(iel) == nnel)
        e(iel,:) = J_mod*B*d(d_dof)-[e_i;e_i;e_i;0;0;0];
    else
        e(iel,:) = J_mod*B*d(d_dof);
    end
end

clear r_S r_F c_S v_S v_F;
end
```

Now, we benchmark the FEM program against a spherically symmetric QD that is infinitely embedded in barrier material. Grundmann *et al.* and Yang *et al.* both solved this problem in spherical coordinates [78, 79] by understanding the symmetry, i.e. the displacement is purely radial with no shear, and application of the boundary condition for the radial displacement $u(r)$ in the QD and barrier materials:

$$u^{-}(R) - u^{+}(R) = \epsilon_0 R \tag{6.59}$$

where R is the radius of the QD and ϵ_0 lattice mismatch of Eq. 6.40, leads to the following strain equations:

$$\epsilon_{rr}(r) = \epsilon_{\theta\theta}(r) = \epsilon_{\phi\phi}(r) = \frac{3K_{QD}\epsilon_0}{3K_{QD} + 4G_B} - \epsilon_0 \quad , r < R \tag{6.60}$$

$$\epsilon_{rr}(r) = \left[\frac{3K_{QD}\epsilon_0}{3K_{QD} + 4G_B}\right]\frac{R^3}{r_3} \quad , r > R \tag{6.61}$$

$$\epsilon_{\theta\theta}(r) = \epsilon_{\phi\phi}(r) = -\left[\frac{3K_{QD}\epsilon_0}{3K_{QD} + 4G_B}\right]\frac{R^3}{2r_3} \quad , r > R \tag{6.62}$$

where K refers to the bulk modulus of the respective material, G refers to the shear modulus of the respective material, and the subscript refers to QD or barrier material. This simple example underscores some important features of strain calculations for QD's: the trace of the strain matrix (dilation) is uniform inside and outside the QD and the strain does not depend on the geometrical size of the QD.

Yang *et al.* solved this problem for the quantum dot/barrier pair of $Si_0.8Ge_{0.2}$/Si and found that inside the QD, strain is -3.4×10^{-3} and dilation is -0.0102 using Eq. 6.62 [79]. Outside the dot, it is trivial to realize that dilation is zero from Eq. 6.62. Our FEM program calculates the strain in Cartesian coordinates and certain operations on the strain matrix give the same result without regard to which orthonormal coordinate system is used to represent the components of strain. One of those invariants is the trace of the strain matrix. Therefore, we compare the trace of the FEM calculated strain matrix to that of the analytically calculated strain matrix. Elastic and lattice constants are taken from Yang and used as inputs to our FEM strain program. The generated mesh used in the calculations consisted of 6137 nodes and 35818 elements, spherical QD of radius 10 units (units are arbitrary), and bounding box with each edge 25 units. Since the displacement must decay to zero at infinity, we artificially placed these boundary conditions on the nodes falling on the surfaces of the bounding box. The QD consisted of 7442 elements and because linear basis functions are used, the strain is constant in each element of the mesh. We found that the average dilation in each element making up the QD was -0.0105 with a standard deviation of 0.00032. Outside the QD in the barrier material, our results were similar.

6.4 Remarks

Due to the QD growth mechanism, strain forms in and around the QD and accurate models describing the electronic structure of the system must include this additional physics. Strain finds its way into the Hamiltonian via deformation theory as a strain potential. In this chapter, we introduced how strain is modeled in the QD using continuum elasticity theory. We deliberately described the theory in matrix notation with the intention of integrating it to the FEM. We then proceeded with the FEM generalities of our approach to continuum elasticity theory, followed by more rigorous derivations of FEM in terms of the stiffness matrix and force vector.

Following the FEM description of strain in a QD system, we offered up how this additional physics could be translated into the Hamiltonian. In very general terms, we described deformation potential theory but delayed the larger necessary discussion of k·p perturbation theory incorporating deformation potential theory and notion of the effective mass for Chap. 7. Finally, we described the FEM program used to solve strain in the QD system and benchmarked the program against the analytic solution of the spherical QD.

Chapter 7: Quantum Dot Intermediate Band Solar Cell Materials

The notion that increasing the efficiency of the conventional solar cell beyond its thermodynamic maximum by rethinking the design based on a simplistic idea could become a reality with the advent of modern technology. The theoretical intermediate band solar cell was offered as a possible solution overcoming the relatively poor efficiency of traditional designs. We found the conventional conversion device absorbs incident light at a maximum efficiency of 31% and could be increased to 41% under fully concentrated light. However, under certain operating conditions, the intermediate band solar cell would increase the maximum conversion efficiency dramatically to 46.8% for incident light and 63.2% under fully concentrated light for one intermediate band. It was also shown that adding another intermediate band incrementally increases the theoretical conversion efficiency to 52.1% for incident light and 72.4% under fully concentrated light. A physical realization of the intermediate band solar cell is the QD-IBSC. In Chap. 3 we detailed the QD-IBSC and found some preliminary combinations of the III-V ternary alloys that could increase the maximum conversion efficiency to 70% based on certain restrictive assumptions.

In this chapter, we look to relax those restrictive assumptions and search for realistic combinations of the technologically important III-V compound semiconductors and their alloys for the QD-IBSC that will increase the thermodynamic efficiency greater than 46% for incident light and greater than 62% for concentrated light. We begin in Sec. 7.1 by evaluating the assumptions that made it possible to find the prospective QD-IBSC materials in Chap. 3.3 and offer alternatives that closer mirror reality. In Sec. 7.2, we describe the model that permits us to determine possible material combinations for the QD-IBSC that boost maximum conversion efficiency beyond the conventional device. Results based on our model are discussed in Sec. 7.3. We make concluding remarks in Sec. 7.4.

7.1 Assumptions

Evaluating the assumptions used to find materials for the QD-IBSC becomes of critical importance to the credibility of making the claim that certain material combinations are better QD-IBSC candidates than others. In order to leverage theoretically models to investigate the electronic structure of S&K grown QD's, assumptions based on experimental observation will help justify model results. In instances where experimental observations are either difficult to quantify or have not been performed, assumptions should be based on a known similar observation, numerical simulations, or some other 'best estimate'. In Chap. 3.3, we assumed the QD was spherically symmetric with a diameter that could be varied. The shape was chosen because of the relative ease in determining the energy levels and respective wavefunctions rather than experimental evidence suggesting a spherical nature to the QD. Reported shapes of QDs are of wide variety, including pyramidal, truncated pyramidal, lens, hemispherical, multifaceted domes, etc. [80, 78, 81, 82, 83, 84]. However, spherical S&K QD's have not been observed. Therefore, the assumption that QD's are spherical should be updated to include more realistic geometry. Although, there is merit to performing calculations on the spherical QD, as it helps us understand some of the more general features and could quite possibly be manufactured at some point in the future due to technological advances.

Due to the complex nature, operating conditions, and various methods of S&K growth, it is difficult to predict the size and shape of the QD's for given material combinations. We performed a literature search to find typical geometry of QD's that will be used in our calculations. Table 7.1 displays a selected few QD systems that have been experimentally observed summarizing typical geometrical shapes and dimensions. It should be noted that growth parameters, such as temperature, have been omitted from Tab. 7.1. They do play a critical role in the size and distribution of the QD's, so there is a certain amount of control during growth.

Table 7.1: Selected experimentally observed data coherent self-assembled S&K growth systems.

Material QD/Barrier	Base (nm^2) or (nm)	Height (nm)	Geometry	Reference
InAs/GaAs	12×12	5-6	pyramid-square base	[85]
(Ga,Al)Sb/GaAs	24-32	2.3-4.1	lens	[86]
InSb/GaAs	55-65	5.3-6.3	lens	[86]
InGaAs/GaAs	15-25×15-25	5-6	pyramid-square base	[87]
InAs/InP	45×45	7	truncated pyramid-square base	[88]
InP/GaInP	60×40	15	truncated pyramid-rectangular base	[89]

Based on reported experimental observations that detail the geometry of the QD, we propose to limit the scope of our analysis to two QD geometries: a pyramid with square base and a truncated pyramid with square base. The scope of geometry is limited to the numerous calculations that must be performed and is justified by the large number of experimental observations that have been reported depicting the proposed geometry. The geometry is highlighted in Fig. 7.1. The pyramid with square base is defined by two dimensions, while the truncated pyramid is defined by three dimensions (see fig. 7.1).

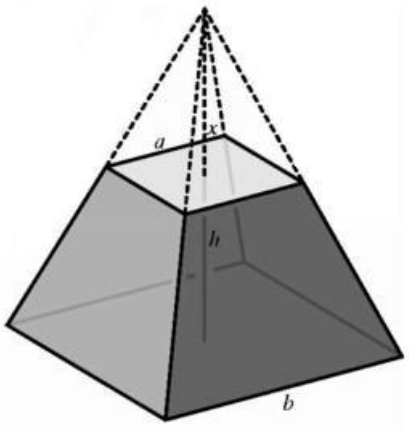


Figure 7.1: The two types of geometry used in our analysis: pyramid and truncated pyramid. The pyramid is defined by the two dimensions b and h on the figure, while the truncated pyramid is defined by three dimensions b, h, and a.

As previously discussed, strain enters the Hamiltonian as an additional potential in self-assembled S&K QD's. This potential, as we will demonstrate, has a profound effect on the band structure and must be included for serious calculations. In Chap. 3.3, we assumed strain was negligible and omitted the strain-induced potential in the Hamiltonian. Similar to the spherically symmetric geometry assumption, strain was assumed negligible because of the relative ease in performing back-of-the-envelop simplistic calculations. In this chapter, we do not assume that strain is negligible and further develop the strain-induced potential in the context of k·p perturbation theory.

For an electron in a periodic potential such as crystal, the wavefunction can be described by Bloch functions [10]:

$$\psi_{n\boldsymbol{k}}(\boldsymbol{r}) = \phi_{n\boldsymbol{k}}(\boldsymbol{r})e^{i\boldsymbol{k}\cdot\boldsymbol{r}} \tag{7.1}$$

where n is the discrete band index, $\mathbf{k}$ is the wavevector, and $\phi_{\mathbf{k}}(\mathbf{r})$ is a periodic function with the periodicity of the crystal. Substituting Eq. 7.1 into Schrödinger's equation, we obtain a equation similar to Schrödinger's equation but with two extra terms:

$$\frac{\partial^2 \psi_{n\boldsymbol{k}}(\boldsymbol{r})}{\partial \boldsymbol{r}^2} = -k^2 e^{i\boldsymbol{k}\cdot\boldsymbol{r}}\phi_{n\boldsymbol{k}}(\boldsymbol{r}) + 2ike^{i\boldsymbol{k}\cdot\boldsymbol{r}}\frac{\partial \phi_{n\boldsymbol{k}}(\boldsymbol{r})}{\partial \boldsymbol{r}} + e^{i\boldsymbol{k}\cdot\boldsymbol{r}}\frac{\partial^2 \phi_{n\boldsymbol{k}}(\boldsymbol{r})}{\partial \boldsymbol{r}^2} \tag{7.2}$$

$$\left[\frac{-\hbar^2}{2m}\cdot\left(-k^2 + 2ik + \nabla^2\right) + V(\boldsymbol{r})\right]\phi_{n\boldsymbol{k}}(\boldsymbol{r}) = E\phi_{n\boldsymbol{k}}(\boldsymbol{r}) \tag{7.3}$$

$$\underbrace{\left[\frac{-\hbar^2\nabla^2}{2m} + V(\boldsymbol{r})\right]}_{H_0}\phi_{n\boldsymbol{k}}(\boldsymbol{r}) + \underbrace{\left[\frac{\hbar^2 k^2}{2m} + \frac{\hbar}{m}\boldsymbol{k}\cdot\boldsymbol{p}\right]}_{H_1}\phi_{n\boldsymbol{k}}(\boldsymbol{r}) = E\phi_{n\boldsymbol{k}}(\boldsymbol{r}) \tag{7.4}$$

where H_0 is considered the unperturbed Hamiltonian, H_1 is considered the perturbed Hamiltonian and we made use of $\mathbf{p} = -i\hbar\nabla$. For direct-band zinc-blende materials, which describe the materials in our research, states near the conduction band minimum ($\mathbf{k} = 0$), $\phi_{\mathbf{k}}(\mathbf{r}) = \phi_0^c(\mathbf{r})$ possess s-like orbital symmetry. States near the valence band maximum possess p-like orbital symmetry and are linear combinations of p_x, p_y, and p_z [90]. At $\mathbf{k} = 0$ and disregarding the spin-orbit coupling for the moment, states are three-fold degenerate in the valence band. We assume that the energy eigenvalues and wavefunctions for $\mathbf{k} = 0$ are known as $E_n(0)$ and $\phi_0^v(\mathbf{r})$. We look for four energy eigenvalues, that of the conduction band and the three valence bands: heavy hole, light hole, and split off band. The corresponding known energy eigenvalues are given as $E_c = E_g$ and $E_{hh} = E_{lh} = E_{so} = 0$, where E_g refers to the direct band gap. Using perturbation theory, we obtain the n^{th} energy eigenvalue

$$E_n(\boldsymbol{k}) \approx E_n(0) + \frac{\hbar^2 k^2}{2m_0} + \frac{\hbar^2}{m_0^2}\sum_{m\neq n}\frac{|\langle n|\boldsymbol{k}\cdot\boldsymbol{p}|m\rangle|^2}{E_n(0) - E_m(0)} \tag{7.5}$$

$$\frac{1}{m_n^*} = \frac{1}{m_0}\left[1 + \frac{2}{m_0 k^2}\sum_{m\neq n}\frac{|\langle n|\boldsymbol{k}\cdot\boldsymbol{p}|m\rangle^2}{E_n(0) - E_m(0)}\right] \tag{7.6}$$

$$E_n(\boldsymbol{k}) = E_n(0) + \frac{\hbar^2 k^2}{2m_n^*} \tag{7.7}$$

where we have used Dirac notation to represent the conduction and valence band basis and the first order matrix elements vanish due to symmetry arguments in the zinc-blende crystals [91]. We quickly observe that the valence bands are coupled to the conduction band through the matrix element $\langle n|\mathbf{k}\cdot\mathbf{p}|m\rangle$, affecting the curvature of the respective band, with the strength of the interaction directly related to E_g. It is seen from Eq. 7.7 that we have recovered the effective mass but in terms of the coupling matrix element.

The valence band basis set that is used in evaluating the coupling matrix is in the JM_j basis by combining the p-like orbitals, $l = 1$, with spin $s = 1/2$ leads to $j = 3/2$ and $j = 1/2$ states. It can be shown that basis set is simple linear combinations [92]

$$|\frac{3}{2},\frac{3}{2}\rangle = \frac{1}{\sqrt{2}}|(x+iy)\uparrow\rangle, \tag{7.8}$$

$$|\frac{3}{2},-\frac{3}{2}\rangle = \frac{1}{\sqrt{2}}|(x-iy)\downarrow\rangle, \tag{7.9}$$

$$|\frac{3}{2},\frac{1}{2}\rangle = \frac{1}{\sqrt{6}}|(x+iy)\downarrow\rangle - \frac{\sqrt{2}}{\sqrt{3}}|z\uparrow\rangle, \tag{7.10}$$

$$|\frac{3}{2},-\frac{1}{2}\rangle = -\frac{1}{\sqrt{6}}|(x-iy)\uparrow\rangle - \frac{\sqrt{2}}{\sqrt{3}}|z\downarrow\rangle, \tag{7.11}$$

$$|\frac{1}{2},\frac{1}{2}\rangle = \frac{1}{\sqrt{3}}|(x+iy)\downarrow\rangle + \frac{1}{\sqrt{3}}|z\uparrow\rangle, \tag{7.12}$$

$$|\frac{1}{2},-\frac{1}{2}\rangle = -\frac{1}{\sqrt{3}}|(x-iy)\uparrow\rangle + \frac{1}{\sqrt{3}}|z\downarrow\rangle, \tag{7.13}$$

$$|\frac{1}{2},\frac{1}{2}\rangle_c = i|s\uparrow\rangle, \tag{7.14}$$

$$|\frac{1}{2},\frac{-1}{2}\rangle_c = i|s\downarrow\rangle \tag{7.15}$$

where we have also included the conduction band states as well. Let us first consider the conduction band, which is a fairly isolated band in most III-V semiconductors [91] and is nondegenerate. In the one band approximation, we apply the basis set to Eq. 7.7 and get an effective mass for electrons in the conduction band

$$\frac{1}{m_c^*} = \frac{1}{m_0}\left[1 + \frac{2}{m_0 k^2}\left(\frac{k^2 P}{2E_g} + \frac{k^2 P}{6E_g} + \frac{k^2 P}{3E_g}\right)\right] \tag{7.16}$$

where $P = \langle s|\mathbf{p}|i\rangle = \langle s|p_i|i\rangle$ (originally defined by Kane [93]), with $i = x, y, z$. This is entirely consistent with our initial derivation of the effective mass as seen in Eq. 3.2-3.5 but gives us additional insight into its nature: the effective mass of electrons in the conduction band increases as the band gap increases. If the spin-orbit interaction is included in the Hamiltonian, the form of Eq. 7.16 changes slightly but since the effective mass is determined experimentally this makes little practical difference. The main assumption in the effective mass equation is that interband separations are large compared with the energies involved in the solution of the effective mass equation. Thus, the effective mass entering in Schrödinger's equation seen in the previous chapters is entirely consistent under the one band or parabolic approximation. However, this is not the case with the degenerate valence band.

As the spin-orbit interaction is included in the Hamiltonian (treated as a perturbation), the degenerate valence band states at $\mathbf{k} = 0$ split into degenerate heavy-hole and light-hole states, and split-off state separated by the energy $\triangle$. The eigenvalues at $\mathbf{k} = 0$ are now given by $E_c = E_g$, $E_{hh} = E_{lh} = 0$, and $E_{so} = -\triangle$. The split-off energy of a particular semiconductor is typically determined experimentally and published in band parameter literature. In our analysis, we make the assumption that the split off band is isolated, as with the conduction band, and does not interact with the other bands [78].

The heavy-hole and light-hole states are degenerate and require degenerate perturbation theory. Using the basis states in the degenerate subspace Eq. 7.8-7.11, we can obtain a 4×4 Hamiltonian that can be diagonalized to determine the dispersion relations. Luttinger and Kohn worked out the Hamiltonian to be [90]

$$H = H_0 + H_1 = \begin{bmatrix} -P+Q & -S & R & 0 \\ -S^* & -P-Q & 0 & R \\ R^* & 0 & -P-Q & S \\ 0 & R^* & S^* & -P+Q \end{bmatrix} \begin{matrix} |\frac{3}{2},\frac{3}{2}\rangle \\ |\frac{3}{2},\frac{1}{2}\rangle \\ |\frac{3}{2},-\frac{1}{2}\rangle \\ |\frac{3}{2},-\frac{3}{2}\rangle \end{matrix} \tag{7.17}$$

$$P = \frac{\hbar^2}{2m_0}\gamma_1\left(k_x^2 + k_y^2 + k_z^2\right) \tag{7.18}$$

$$Q = \frac{\hbar^2}{2m_0}\gamma_2\left(k_x^2 + k_y^2 - 2k_z^2\right) \tag{7.19}$$

$$R = \frac{\hbar^2}{2m_0}\sqrt{3}\left[-\gamma_2\left(k_x^2 - k_y^2\right) + 2i\gamma_3 k_x k_y\right] \tag{7.20}$$

$$S = \frac{\hbar^2}{2m_0}2\sqrt{3}\gamma_3\left(k_x - ik_y\right)k_z \tag{7.21}$$

where γ_1, γ_2, and γ_3 are called the modified Luttinger parameters. These parameters contain P, that couples the conduction band to valence bands, and are convenient to work with theoretically

although they don't have any physical meaning. Similar to the effective mass, Luttinger parameter values are found in literature containing band parameters.

In the Hamiltonian equation above, we can replace the k-values by $k_j = i \cdot \nabla_j$ that will result in four coupled differential equations. This is a much different scenario than the single effective mass differential equation, i.e. Schrödinger's equation with spatially varying effective mass, describing the conduction band electrons. For our analysis, we are only concerned with the valence band maximum in both the barrier and QD materials, instead of locating bound states caused by the valence band offset. As noted in Chap. 3, valence band offsets will support bound states, in fact many bound states, and as a criterion for potential QD-IBSC materials, only material combinations with negligible valence band offsets were considered. However, when strain is introduced, the likely hood of this condition is greatly diminished because the valence band offset caused by a difference in energy gaps would need to negate the strain potential caused by the lattice mismatch. In addition, the valence band offset is uniform but due to the non-uniform strain for QD geometry considered, the strain potential will be non-uniform furthering the difficulty to satisfy this criterion. For the purposes of this analysis, we make the assumption that bound states formed in any valence band offset will merge together as they spread out into bands, effectively shrinking the band gap of the barrier material. The modified band gap of the barrier material will be determined by the following formula

$$E_g = E_g^B - (V_o + V_s) \tag{7.22}$$

where E_g^B is the barrier's unmodified direct band gap at $k = 0$, V_o is the valence band offset caused by difference in energy gaps at $k = 0$, and V_s is the strain potential occurring at the valence band at $k = 0$.

Returning to the 4×4 Hamiltonian, we can solve for the dispersion relations by diagonalizing Eq. 7.17

$$|H_{ij} - \delta_{ij} E| = 0 \tag{7.23}$$

and find that the degeneracy of the heavy hole and light hole is lifted for $k \neq 0$

$$E_{hh} = -P + \sqrt{Q^2 + R^* \cdot R + S^* \cdot S} \tag{7.24}$$

$$E_{lh} = -P - \sqrt{Q^2 + R^* \cdot R + S^* \cdot S} \tag{7.25}$$

with E_{hh} and E_{lh} each being doubly degenerate. We will now build on the Hamiltonian in Eq. 7.17 to incorporate the strain induced potential. In the last chapter, we found how strain introduces an additional potential in Schrödinger's equation via deformation potential theory. The strain potential from Eq. 6.53 is treated as a perturbation using the k·p Hamiltonian basis, leaving only the terms linear in strain. Pikus and Bir first derived this strain potential in this context and, as expected, has a very similar form as Eq. 7.17 [94, 95, 96]

$$V_s = \begin{bmatrix} -p+q & -s & r & 0 \\ -s^* & -p-q & 0 & r \\ r^* & 0 & -p-q & s \\ 0 & r^* & s^* & -p+q \end{bmatrix} \begin{matrix} |\frac{3}{2}, \frac{3}{2}\rangle \\ |\frac{3}{2}, \frac{1}{2}\rangle \\ |\frac{3}{2}, -\frac{1}{2}\rangle \\ |\frac{3}{2}, -\frac{3}{2}\rangle \end{matrix} \tag{7.26}$$

$$p = a_v \left(\epsilon_{xx} + \epsilon_{yy} + \epsilon_{zz}\right) \tag{7.27}$$

$$q = -\frac{b}{2} \left(\epsilon_{xx} + \epsilon_{yy} - 2\epsilon_{zz}\right) \tag{7.28}$$

$$r = \frac{\sqrt{3}b}{2} \left(\epsilon_{xx} - \epsilon_{yy}\right) - id\epsilon_{xy} \tag{7.29}$$

$$s = -d \left(\epsilon_{xz} - i\epsilon_{yz}\right) \tag{7.30}$$

where the three parameters a_v, b, and d are the deformation constants as introduced in Chap. 6.2. Understanding that these constants arise from perturbation theory, it can be inferred that they describe the coupling of the valence bands to the strain [96]. Theoretically, it is possible to calculate

them but, as with the effective mass, it is more convenient to fit the deformation potentials to experimental results. We can immediately see the relationship between Eq. 7.17 and Eq. 7.26

$$k_i k_j \longrightarrow \epsilon_{ij} \tag{7.31}$$

$$-\frac{\hbar^2}{2m_0}\gamma_1 \longrightarrow a_v \tag{7.32}$$

$$-\frac{\hbar^2}{m_0}\gamma_2 \longrightarrow b \tag{7.33}$$

$$-\frac{\sqrt{3}\hbar^2}{m_0}\gamma_3 \longrightarrow d \tag{7.34}$$

Now this strain potential Hamiltonian is inserted into the Luttinger and Kohn Hamiltonian such that Eq. 7.17 is modified to

$$H_{PB} = \begin{bmatrix} -P-p+Q+q & -S-s & R+r & 0 \\ -S^*-s^* & -P-p-Q-q & 0 & R+r \\ R^*+r^* & 0 & -P-p-Q-q & S+s \\ 0 & R^*+r^* & S^*+s^* & -P-p+Q+q \end{bmatrix} \tag{7.35}$$

and we can solve for the dispersion relations by diagonalizing H_{BP}, which results in

$$E_{hh} = -\frac{\hbar^2}{2m_0}\gamma_1\left(k_x^2+k_y^2+k_z^2\right) - a_v\left(\epsilon_{xx}+\epsilon_{yy}+\epsilon_{zz}\right) + \sqrt{A+B+C} \tag{7.36}$$

$$E_{lh} = -\frac{\hbar^2}{2m_0}\gamma_1\left(k_x^2+k_y^2+k_z^2\right) - a_v\left(\epsilon_{xx}+\epsilon_{yy}+\epsilon_{zz}\right) - \sqrt{A+B+C} \tag{7.37}$$

where

$$A = \left(\frac{\hbar^2}{m_0}\gamma_2\right)^2 \cdot \left(k_x^2+k_y^2+k_z^2\right)^2 + 3\left(\frac{\hbar^2}{m_0}\right)^2\left[(\gamma_3)^2-(\gamma_2)^2\right]\cdot\left(k_x^2k_y^2+k_x^2k_z^2+k_y^2k_z^2\right) \tag{7.38}$$

$$B = \frac{b^2}{2}\left[(\epsilon_{xx}-\epsilon_{yy})^2+(\epsilon_{xx}-\epsilon_{zz})^2+(\epsilon_{yy}-\epsilon_{zz})^2\right] + d^2(\epsilon_{xy}^2+\epsilon_{xz}^2+\epsilon_{yz}^2) \tag{7.39}$$

$$C = -b\frac{\hbar^2}{m_0}\gamma_2\left[3\left(k_x^2\epsilon_{xx}+k_y^2\epsilon_{yy}+k_z^2\epsilon_{zz}\right)-\left(k_x^2+k_y^2+k_z^2\right)\left(\epsilon_{xx}+\epsilon_{yy}+\epsilon_{zz}\right)\right] - d2\sqrt{3}\frac{\hbar^2}{m_0}\gamma_3\left(k_xk_y\epsilon_{xy}+k_xk_z\epsilon_{xz}+k_yk_z\epsilon_{yz}\right) \tag{7.40}$$

Again, each eigenvalue is doubly degenerate but we find that the strain lifts the degeneracy between of the light hole and heavy hole at $k = 0$

$$E_{hh}(k=0) = -a_v\left(\epsilon_{xx}+\epsilon_{yy}+\epsilon_{zz}\right) + B^{1/2} \tag{7.41}$$

$$E_{lh}(k=0) = -a_v\left(\epsilon_{xx}+\epsilon_{yy}+\epsilon_{zz}\right) - B^{1/2} \tag{7.42}$$

Values for the deformation potential constants are such that $a_v < 0$, $b < 0$, and $d < 0$ and for compressive strain, as in the QD, the valence band moves down in energy (and conduction band moves up in energy). Under these conditions, the heavy hole energy at $k = 0$ sits energetically higher than the light hole at $k = 0$ and will be used as the strain potential V_s in Eq. 7.22 to calculate the barrier's band gap with strain included.

We must note that bound states located within a potential well formed by the valence bands of a QD heterostructure could be solved using FEM. We follow a similar procedure from Chap. 4. We must first construct the appropriate action integrals that will generate the coupled "Schrödinger equations" of Eq. 7.35. Next, we choose an appropriate basis functions for each the four elemental wavefunctions (light hole, heavy hole, and their spin degeneracies) using the previously developed

prescription. Then, the elemental wavefunctions are substituted into the action integrals. The integrals are carried out to construct the elemental matrix and then properly overlaid into the global matrix. We then exercise the principle of station action by placing a variation on the unknown nodal wavefunction values. Finally, we treat the boundary conditions within the global matrix and it is input into any standard generalized eigenvalue solver. Eigenvalues represent potential bound states and corresponding eigenvectors represent nodal wavefunction values.

7.2 Model

The theoretical model that will be used to investigate promising QD-IBSC materials begins by defining the geometry. We have discussed the two types of geometry that will be used in the calculations: a pyramid with square base and a truncated pyramid with square base. The three dimensional mesh program introduced in Chap. 5 will be used to tessellate the stated geometry into tetrahedral elements. Next, we calculate the strain for chosen QD and barrier materials using the finite element method and program outlined in Chap. 6. As a reference and guide for material selections, Fig. 7.2 depicts the direct bandgaps for the III-V binary compound semiconductors (points) and some of their ternary alloys (curves) as a function of the lattice constant. An observation

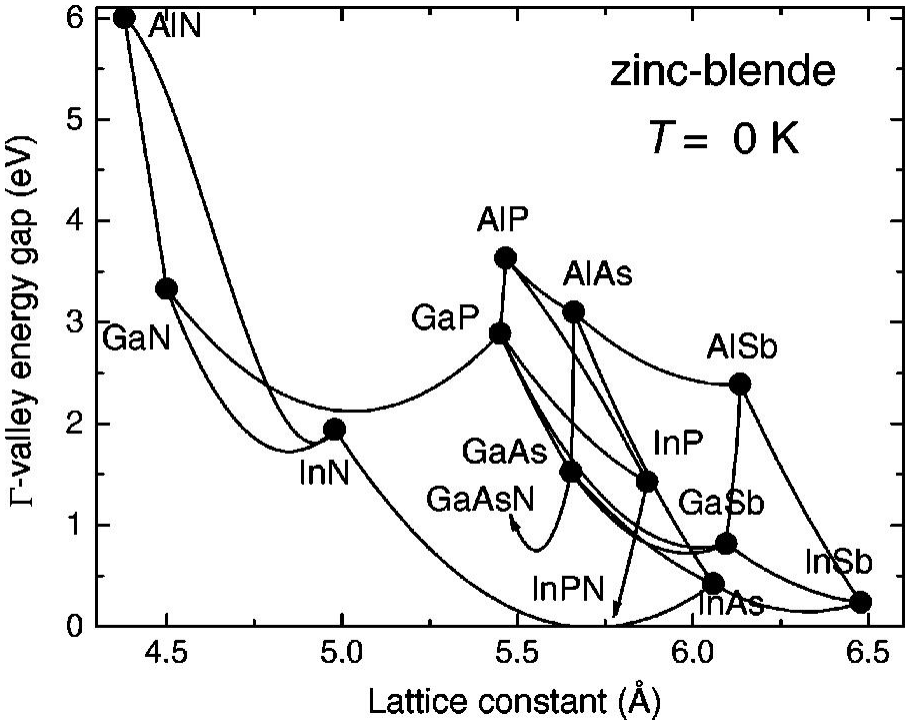


Figure 7.2: Direct band gaps ($k = 0$) for the III-V binary compound semiconductors (points) and some of their ternary allows (curves) as a function of their lattice constants. In general, QD material will have a larger lattice constant than barrier counterparts. Courtesy of [1].

made from the figure is the lattice constants of small bandgap semiconductors are generally greater than large bandgap semiconductors. This is the reason why most QD's experience compressive strain, which in-turn leads to the widening of the QD band gap. The resulting strain and deformation constants are then used to calculate the respective strain potential within the QD/barrier material system. The heavy hole strain potential, Eq. 7.41, will be used to calculate the shift in the valence band due to strain and the conduction band strain potential, Eq. 6.54, will be used to calculate the shift in the conduction band due to strain. The Hamiltonian consisting of a spatially varying effective mass and the conduction band offset potential plus the conduction band strain potential

is used in the FEM context described in Chap. 4 to calculate the bound states appearing in the conduction band, which will be performed on the same mesh used for the strain calculations. As mentioned, the valence band offset is assumed to merge with the barrier's valence band, essentially shrinking the barrier's band gap. This assumption will be valid for 'small' valence band offsets or ≈ 50 meV. This value is chosen based on numerical experiments performed on the spherical QD. Figure 7.3 depicts the conduction and valence band offsets for the direct band gap III-V binary semiconductors. The conduction (valence) band offset between two semiconductors is taken as the difference between the conduction (valence) energetic position.

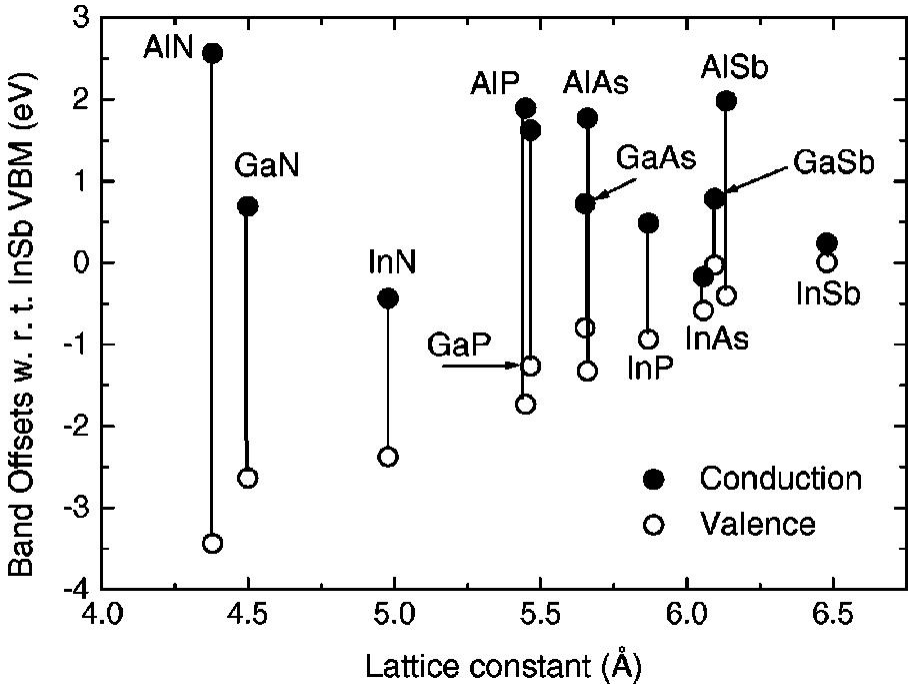


Figure 7.3: Conduction (filled) and valence (open) band offsets for the III-V binary compound semiconductors as a function of their lattice constants. The direct band gap for a given semiconductor corresponds to the difference between the conduction and valence band positions. Courtesy of [1].

We have decoupled the valence band from the conduction band for the purpose of performing a large number of numerical calculations, thereby reducing the computational cost. This is appropriate due to weak interaction between the two bands [78]. In addition, it has been suggested that more complicated Hamiltonians, i.e. coupling the valence to the conduction band (8×8 Hamiltonian) or including additional physics (piezoelectric effect, where solutions are affected by less than 1 meV [78, 97]), give results that are no better than the input band parameters used [98]. Instead of looking for materials to produce the maximum efficiency, as in Chap. 3, we will be choosing slightly less efficient systems that are more robust under parameter perturbation to account for uncertainties. As in our previous work from Chap. 3, band parameters are taken from the Vurgaftman *et al.* comprehensive review for the III-V semiconductors. A particular material system will then be compared to the theoretical models outlined in Chap. 2. Only those material systems that match the IBSC efficiency greater than 46% for unconcentrated light and greater than 62% for fully concentrated light will be identified as potential candidates.

For the QD-IBSC containing one intermediate band under unconcentrated light, the effective band gap must be in the range of 2.06 eV $\leq E_g \leq$ 2.71 eV for an efficiency $\geq$ 46% (see Sec. 2.1). Under fully concentrated light, the effective band gap must in the range of 1.63 eV $\leq E_g \leq$ 2.31 eV for an efficiency $\geq$ 62%. Taken together, we have the requirement 1.63 eV $\leq E_g \leq$ 2.71 eV

for the effective band gap. This begins to narrow down the potential barrier materials from Fig. 7.2 to the binary compounds: AlSb and InN and ternary alloys: AlGaAs, AlInAs, GaInP, AlInP, InAsSb, AlAsSb, GaAsP, GaPSb, AlPSb, GaInN, AlInN, GaAsN, GaPN, and InPN. In addition to this requirement placed on the effective band gap, we place a requirement on the transition energy E_2 to account for design uncertainties

$$E_2(E_1) \approx \frac{b(E_1) + a(E_1)}{2} \tag{7.43}$$

where $a(E_1)$ and $b(E_1)$ are the polynomial functions taken from Eqs. 2.13 and 2.14 (not to be confused with the deformation potential constants). Potential QD materials must possess a gap that is smaller than the barrier gap, thereby limiting the potential QD's to the binary compounds: GaSb, InSb, and InAs and all their ternary alloys.

As a practical matter, the efficiency of a particular system does not depend on the order of the energy transition levels, E_1 and E_2. Efficiency depends only on the transition energies themselves. As an example, we saw in Chap. 2 that a theoretical efficiency of 63.2% is achieved under fully concentrated light when $E_1 = 0.70$ eV and $E_2 = 1.23$ eV. However, the efficiency of 63.2% is also achieved if $E_1 = 1.23$ eV and $E_2 = 0.70$ eV rendering the order of the transition energies unimportant. With this understanding, we place a criterion on the conduction band offset of potential material systems. The conduction band offset must be at a minimum 0.57 eV, which corresponds to the smallest energy transition in our search (see Eq. 2.14).

For the QD-IBSC containing two intermediate bands under unconcentrated light, we search for conversion efficiencies $\geq 50\%$. For fully concentrated light, we search for for conversion efficiencies $\geq 70\%$. We do know that the effective band gap must be ≥ 2.0 eV in order to produce desired efficiencies, which will eliminate 6 of the 12 binary compound semiconductors from being considered barrier material. Potential barrier materials include: AlN, GaN, GaP, AlP, AlAs, and AlSb with their ternary alloys considered candidates as well. Unfortunately, similar requirements placed on the energy transitions that aid in the material selection process for the QD-IBSC containing one intermediate band are not available for the QD-IBSC containing two intermediate bands. Each effective band gap has its own efficiency profile in terms of the two energies E_1 and E_2. The analytic expressions describing a desired interval for one effective band gap will not be the same for another effective band gap, making this type of calculation less useful. A remedy is to perform a sensitivity analysis on each effective band gap of those material selections that are found to meet the efficiency requirement. Potential QD materials will be the the same as in the one intermediate band search and a similar criterion placed on the conduction band offset will apply.

We apply our theoretical model to InAs QDs embedded in GaAs, which is very widely studied from a experimental and theoretical standpoint, for benchmarking purposes. Band parameters for InAS and GaAs are taken from Vurgaftman *et. al* and for convenience displayed in Tab. 7.2. We immediately notice that the valence band offset is non-negligible, 0.21 eV, thus immediately eliminating the material system from being a potential QD-IBSC. In any case, the conduction band offset is 0.892 eV and the large lattice mismatch of 7.16% is more than sufficient to stimulate S&K growth. We choose the square-based pyramid geometry that has the dimensions 16×3 nm (base $\times$ height) for InAS to benchmark the calculations performed on this system by Kuo *et al.* [74]. The analysis was completed on a cubic mesh defined by the dimension of 40 nm consisting of over $3,000$ nodes and $20,000$ elements. We calculated the average elemental conduction band strain potential inside the QD as 0.306 eV using Eq. 6.54, which increases the conduction band of InAs by the same amount, reducing the conduction band offset by about 0.586 eV (average elemental conduction band strain potential outside the QD is small). Using Eq. 7.41 to calculate the heavy hole strain potential, we find that the average elemental valence band strain potential inside the QD is 0.16 eV for the InAs band gap increase of 0.146 eV. Kou *et al.* calculated the conduction band strain potential to be 0.313 eV and an InAs band gap increase of 0.142 eV, confirming our strain calculations. We found one bound state located at 1.352 eV with respect to the unstrained valence band of GaAs and a double degenerate bound state located at 1.492 eV. Kou *et al.* has similar results: one bound state located at 1.354 eV and a double degenerate bound state located at 1.495 eV. In addition to

Table 7.2: Selected band parameters for InAs and GaAs.

Parameter	InAs	GaAs
a_{LC} (nm)	0.565325	0.60583
E_g (eV)	1.519	0.417
m_e^*	0.067	0.026
VBO (eV)	−0.80	−0.59
a_c (eV)	−7.17	−5.08
a_v (eV)	−1.16	−1.00
b (eV)	−2.0	−1.8
d (eV)	−4.8	−3.6
C_{11} (GPa)	1221	832.9
C_{12} (GPa)	566	452.6
C_{44} (GPa)	600	395.9

the conduction band bound states, Kou *et al.* calculated the hole bound states and compared the calculated fundamental transition energy ($E_{e0} \rightarrow E_{h0}$) to experimental photoluminescence spectra in good agreement.

7.3 Results

The calculations were made on six buried QDs, two for each type of geometry, each labeled structure A, B, C, D, E, and F and summarized in Tab. 7.3. The dimensions of structures C and D are selected based on a similar in volume to the QD materials found in Sec. 3.3. The dimensions of structures A and B are chosen to have dimensions 3/4 that of structures C and D, while the dimensions of structures E and F are chosen to have dimensions 4/3 that of structures C and D. This will give us a good indication as to how potential materials for the QD-IBSC respond to changes in size and will help bound our analysis.

Table 7.3: The geometric properties of the QD structures used in this analysis. The geometric properties b, h, and a refer to the dimensions in Fig. 7.1, while r refers to a sphere of radius r having the same volume as the respective structure. Additionally, we include the volume of each structure. Structures C and D have a similar volume to those QDs found in Chap. 3, while structures A,B,E, and F are based on the dimensions of structures C and D to help bound the analysis.

Geometric Property	A	B	C	D	E	F
Shape	pyramid	truncated pyramid	pyramid	truncated pyramid	pyramid	truncated pyramid
b (nm)	7.5	6	10	8	13.3	10.6
h (nm)	4.5	4.5	6	6	8	8
a (nm)	-	3.75	-	5	-	6.66
Volume (nm^3)	84.4	108.8	200	258	471.7	606.2
r (Å)	27.2	29.6	36.3	39.5	48.3	52.5

Bandgaps in semiconductors are temperature dependent quantities that are often empirically fitted to the Varshni function [99]

$$E_g(T) = E_g(0) - \frac{\alpha T^2}{T + \beta} \tag{7.44}$$

where α and β are adjustable Varshni parameters found in semiconductor band parameter tables. For the QD-IBSC, it is assumed that the device will be operating at ambient temperature and the bandgaps are adjusted accordingly. To get a sense of how the significant temperature affects the bandgap, consider the binary semiconductor GaAs, which has a direct bandgap of 1.519 at 0°K. The Varshni parameters are taken as $\alpha = 0.5405$ meV/K and $\beta = 204$ K, and from Eq. 7.44 the bandgap of GaAs is adjusted to 1.42 at 300°K. For all ternary alloys, the bandgap is assumed to fit the simple quadratic form [100]

$$E_g(A_{1-x}B_x) = (1-x)E_g(A) + xE_g(B) - x(1-x)C \tag{7.45}$$

where x is the molar concentration ($0 \leq x \leq 1$) and C is called the bowing parameter. The bowing parameter accounts for the deviation from linear interpolation between the two binary semiconductors. In general, the bowing parameter for the III-V semiconductors is positive meaning the bandgap is smaller than would be if linear interpolation was used. So to derive the bandgap of a ternary alloy, we first correct the binary semiconductor for temperature and then calculate its bandgap using Eq. 7.45.

We now give a description of how the model outlined from Sec. 7.2 was executed followed by a discussion of the results. First, we call the FEM preprocessing program from Chap. 5 to generate the mesh for the respective geometry of the material system. The output arguments are stored for use as inputs to the FEM processing program. Next, we run a MATLAB script calling the various FEM processing and post-processing programs to produce a formatted document containing the prospective material systems for the given geometry. The script scans through all the QD and barrier permutations (molar concentrations are increased by 0.02 for $0 \leq x \leq 1$), filters the material systems that satisfy the design rules, calculates the strain potential using the function *FEM_3D_Strain* from Chap. 6, calculates the energy eigenvalues and eigenvectors using the processing/post processing programs from Chap. 5, and writes to an external file only those material properties that support one or two energy levels. Material properties are then compared to the IBSC theoretical model and only those that match the desired efficiencies are considered a potential QD-IBSC. We should note that the band parameters of ternary alloys are interpolated using Eq. 7.45 for a given molar concentration x. For those parameters that do not have C, where linear interpolation is sufficient, we use the value $C = 0$ in the equation [1].

In Tabs. 7.4 - 7.9, we show specific potential QD-IBSC material systems found for each QD structure that satisfy the stated design criteria. In each table, we display the barrier/QD material, the independent energy transition levels E_i, the lattice mismatch $\triangle_{LC}$, the efficiency threshold the system meets η, and the concentration factor X. Due to the symmetry of the QD for both geometries, the first and second excited energy levels are double degenerate, so all the material systems found support either one or three energy levels. As a practical matter, this helps us speak qualitatively to what material systems might be potential QD-IBSCs (see Fig. 7.10 below and accompanying discussion).

In Tab. 7.4, we notice that all the potential QD-IBSC material systems for structure A would be designed to operate for fully concentrated light. We found that although many of material systems supported either one or three energy levels, energy transitions E_1 or E_2 were too large for barrier materials suitable for the $X = 1$ QD-IBSC design. We also observed, in general, that material systems supporting three energy transitions with band gaps ideal for $X =$ achieved efficiencies between $60 - 65\%$, thereby precluding them from consideration due to our high efficiency criterion. In terms of the total number of material systems supporting one or three energy levels, structure A had the most when compared to its two pyramid counterparts.

The potential QD-IBSC materials for structure B are displayed in Tab. 7.5. We immediately observe the number of potential QD-IBSC materials meeting the efficiency criterion is two and designed for unconcentrated light. This is in contrast to structure A, which is of similar volume. A comparison with structure A indicates that the energy states are more tightly bound and in many cases, in material systems that support a single energy level for the geometry of structure A, the same material systems in structure B support three levels. As might be expected, the total amount of material systems supporting one or three energy levels is similar to structure A and has the most

Table 7.4: Potential QD-IBSC material systems that produce the desired efficiency, η, for structure A under unconcentrated, $X = 1$, and/or fully concentrated light, $X = 1/F_{sun}$. The energy transitions E_1, E_2, and E_3 refer to those in Fig. 2.1. The lattice mismatch between the QD and barrier material is designated as $\triangle_{LC}$.

Barrier/QD	E_1 (eV)	E_2 (eV)	E_3 (eV)	$\triangle_{LC}(\%)$	$\eta(\%)$	X
$Al_{0.68}In_{0.32}As/InAs_{0.82}N_{0.18}$	1.22	0.71	-	1.31	≥ 62	$1/F_{sun}$
$Al_{0.66}In_{0.34}As/InAs_{0.82}N_{0.18}$	1.21	0.69	-	1.17	≥ 62	$1/F_{sun}$
$Al_{0.64}In_{0.36}As/InAs_{0.84}N_{0.16}$	1.16	0.66	-	1.40	≥ 62	$1/F_{sun}$
$Al_{0.64}In_{0.36}As/InAs_{0.82}N_{0.18}$	1.20	0.66	-	1.03	≥ 62	$1/F_{sun}$
$Al_{0.62}In_{0.38}As/InAs_{0.84}N_{0.16}$	1.15	0.63	-	1.26	≥ 62	$1/F_{sun}$
$Al_{0.76}In_{0.24}As/InP_{0.92}N_{0.08}$	1.26	0.75	0.22	0.73	≥ 70	$1/F_{sun}$
$Ga_{0.40}In_{0.60}P/InP_{0.90}N_{0.10}$	1.12	0.61	-	1.38	≥ 62	$1/F_{sun}$
$GaAs_{0.50}P_{0.50}/InP_{0.90}N_{0.10}$	1.26	0.71	-	4.12	≥ 62	$1/F_{sun}$
$GaAs_{0.48}P_{0.52}/InP_{0.90}N_{0.10}$	1.26	0.74	-	4.20	≥ 62	$1/F_{sun}$

compared to its two truncated pyramid counterparts.

Table 7.5: Potential QD-IBSC material systems that produce the desired efficiency, η, for structure B under unconcentrated, $X = 1$, and/or fully concentrated light, $X = 1/F_{sun}$. The energy transitions E_1, E_2, and E_3 refer to those in Fig. 2.1. The lattice mismatch between the QD and barrier material is designated as $\triangle_{LC}$.

Barrier/QD	E_1 (eV)	E_2 (eV)	E_3 (eV)	$\triangle_{LC}(\%)$	$\eta(\%)$	X
$Al_{0.50}In_{0.50}P/InAs_{0.56}N_{0.44}$	1.00	1.55	-	1.49	≥ 46	1
$AlP_{0.70}Sb_{0.30}/InAs_{0.56}N_{0.44}$	1.02	1.57	-	1.48	≥ 46	1

The potential QD-IBSC materials for structure C are displayed in Tabl. 7.6. There are three potential QD-IBSC materials that meet all the design criteria for unconcentrated light. We observe that the potential QD-IBSC material systems are similar to that of structure B. This is interesting because the two structures have different geometries, which suggests that the material systems are robust in terms of it acting as QD-IBSC. The material system AlAsSb/InAsN that appears in Tab. 7.6 would appear in Tab. 7.5 if the efficiency criterion was 45%, further confirming the similarities between the two structures.

Table 7.6: Potential QD-IBSC material systems that produce the desired efficiency, η, for structure C under unconcentrated, $X = 1$, and/or fully concentrated light, $X = 1/F_{sun}$. The energy transitions E_1, E_2, and E_3 refer to those in Fig. 2.1. The lattice mismatch between the QD and barrier material is designated as $\triangle_{LC}$.

Barrier/QD	E_1 (eV)	E_2 (eV)	E_3 (eV)	$\triangle_{LC}(\%)$	$\eta(\%)$	X
$Al_{0.50}In_{0.50}P/InAs_{0.56}N_{0.44}$	1.54	1.00	-	1.49	≥ 46	1
$AlAs_{0.80}Sb/InAs_{0.84}N_{0.16}$	1.05	1.63	-	2.25	≥ 46	1
$AlP_{0.70}Sb_{0.30}/InAs_{0.56}N_{0.44}$	1.01	1.57	-	1.48	≥ 46	1

The potential QD-IBSC materials for structure D are displayed in Tab. 7.7. We notice that there are two potential QD-IBSC materials that could be designed for both a concentration factor of $X = 1$ and $X = 1/F_{sun}$, indicating the flexibility of the materials. This is the first occurrence of

potential QD-IBSCs that could operate at the optimal efficiency levels of both concentration factors.

Table 7.7: Potential QD-IBSC material systems that produce the desired efficiency, η, for structure D under unconcentrated, $X = 1$, and/or fully concentrated light, $X = 1/F_{sun}$. The energy transitions E_1, E_2, and E_3 refer to those in Fig. 2.1. The lattice mismatch between the QD and barrier material is designated as $\triangle_{LC}$.

Barrier/QD	E_1 (eV)	E_2 (eV)	E_3 (eV)	$\triangle_{LC}(\%)$	$\eta(\%)$	X
$AlP_{0.36}Sb_{0.64}/InAs_{0.82}N_{0.18}$	0.78	1.34	-	0.52	≥ 46	1
$AlP_{0.36}Sb_{0.64}/InAs_{0.82}N_{0.18}$	0.78	1.34	-	0.52	≥ 62	$1/F_{sun}$
$AlP_{0.34}Sb_{0.66}/InAs_{0.82}N_{0.18}$	0.80	1.32	-	0.75	≥ 62	$1/F_{sun}$
$AlP_{0.32}Sb_{0.68}/InAs_{0.84}N_{0.16}$	0.79	1.33	-	0.61	≥ 46	1
$AlP_{0.32}Sb_{0.68}/InAs_{0.84}N_{0.16}$	0.79	1.33	-	0.61	≥ 62	$1/F_{sun}$
$GaP_{0.90}N_{0.10}/InAs_{0.52}N_{0.48}$	0.93	1.49	-	3.46	≥ 46	1
$GaP_{0.88}Sb_{0.12}/InAs_{0.52}N_{0.48}$	0.88	1.46	-	3.83	≥ 46	1

The potential QD-IBSC materials for structure E are displayed in Tab. 7.8. Similar to structure D, there are two potential QD-IBSC materials that could be designed for both a concentration factor of $X = 1$ and $X = 1/F_{sun}$. The AlPSb/InAsN combination is most flexible in the sense that its design is suitable for both concentration factors and both structures D and E. The material combination GaPN/InAsN also appears in structure D indicating geometric flexibility. We should note that the total number of material systems satisfying the design criteria was much less than its two geometric counterparts structure A and C.

Table 7.8: Potential QD-IBSC material systems that produce the desired efficiency, η, for structure E under unconcentrated, $X = 1$, and/or fully concentrated light, $X = 1/F_{sun}$. The energy transitions E_1, E_2, and E_3 refer to those in Fig. 2.1. The lattice mismatch between the QD and barrier material is designated as $\triangle_{LC}$.

Barrier/QD	E_1 (eV)	E_2 (eV)	E_3 (eV)	$\triangle_{LC}(\%)$	$\eta(\%)$	X
$AlAs_{0.78}Sb_{0.22}/InAs_{0.84}N_{0.16}$	0.90	1.38	0.42	2.08	≥ 50	1
$AlAs_{0.78}Sb_{0.22}/InAs_{0.84}N_{0.16}$	0.90	1.38	0.42	2.08	≥ 70	$1/F_{sun}$
$AlP_{0.36}Sb_{0.64}/InAs_{0.82}N_{0.18}$	0.79	1.33	-	0.52	≥ 46	1
$AlP_{0.36}Sb_{0.64}/InAs_{0.82}N_{0.18}$	0.79	1.33	-	0.52	≥ 62	$1/F_{sun}$
$AlP_{0.34}Sb_{0.66}/InAs_{0.82}N_{0.18}$	0.80	1.32	-	0.75	≥ 62	$1/F_{sun}$
$GaP_{0.94}N_{0.06}/InAs_{0.56}N_{0.44}$	0.99	1.54	-	3.52	≥ 46	1
$GaP_{0.92}N_{0.08}/InAs_{0.56}N_{0.44}$	0.94	1.50	-	3.89	≥ 46	1
$GaP_{0.92}N_{0.08}/InAs_{0.54}N_{0.46}$	0.95	1.54	-	3.49	≥ 46	1
$GaP_{0.88}N_{0.12}/InAs_{0.54}N_{0.46}$	0.86	1.44	-	4.23	≥ 46	1

The potential QD-IBSC materials for structure F are displayed in Tab. 7.8. We notice that the material combinations are a subset of the material combinations found in structure A. The difference being the energy level in structure F is more tightly bound such that the energy transitions E_1 and E_2 are now opposite of structure A. This larger structure has taken advantage of the flexibility of the IBSC to accommodate energy transition permutations. Like structure E, structure F did not have many material systems satisfying the design criteria. The reason is as the QD increases in size, it will support more energy levels thereby limiting the potential QD-IBSCs. Although not stated explicitly in the previous chapters but implied, we limit the QD-IBSC to one or two energy levels[1]

[1]The QD-IBSC could support more energy levels if one of the energy levels is degenerate, as we saw in our analysis.

because they are less likely to merge with one another or the conduction band. In addition, more energy levels add complexity to the device and increases the likely-hood it will not operate under the ideal IBSC conditions (i.e. non-radiative recommendation). The reasoning is similar to why we look for negligible valence band offsets.

Table 7.9: Potential QD-IBSC material systems that produce the desired efficiency, η, for structure F under unconcentrated, $X = 1$, and/or fully concentrated light, $X = 1/F_{sun}$. The energy transitions E_1, E_2, and E_3 refer to those in Fig. 2.1. The lattice mismatch between the QD and barrier material is designated as $\triangle_{LC}$.

Barrier/QD	E_1 (eV)	E_2 (eV)	E_3 (eV)	$\triangle_{LC}(\%)$	$\eta(\%)$	X
$Al_{0.68}In_{0.32}As/InAs_{0.82}N_{0.18}$	0.68	1.24	-	1.31	≥ 62	$1/F_{sun}$
$Al_{0.66}In_{0.34}As/InAs_{0.82}N_{0.18}$	0.68	1.20	-	1.17	≥ 62	$1/F_{sun}$
$Al_{0.64}In_{0.36}As/InAs_{0.84}N_{0.16}$	0.65	1.16	-	1.40	≥ 62	$1/F_{sun}$

In the tables above, we considered all the III-V ternary alloys and did not dismiss any material system that is considered highly developmental. The aspiration is to find a material breakthrough that will spur additional research on the QD-IBSC design rather than only search those material systems that could be manufactured today. The potential material systems found should focus the research of the QD-IBSC to those combinations and spur technological advances in QD growth. Nevertheless we briefly discuss some previous work with InAsN, which appears to be very important in designing a QD-IBSC device, to illustrate that the ternary has been grown as a QD. Bais *et al.* produced InAsN QDs embedded in GaAs and GaAsN engineered to emit at 1.3 μm at room temperature [101]. Danitsev *et al.* produced InAsN QDs in GaAs and measure an average height of 2.2 nm [102]. Schumann *et al.* produced InAsN QDs with nitrogen concentrations $0.00 - 0.043$ [103]. They found that as the amount of nitrogen increases, the QDs increase in size as well.

Although we found potential QD-IBSC materials satisfying the design criteria and in some cases exhibiting flexibility across the QD structures, material systems are QD size dependent. In addition, we put a strict efficiency design criterion that the material systems must meet in order to be considered. As a practical matter, material systems will most likely not be grown to the exact dimensions of the structures, and we might decide to select a material system that has a slightly less design efficiency than our efficiency criterion. In response, we show an exhaustive list of potential barrier/QD combinations in Tab. 7.10 that satisfied all but the efficiency design considerations and only support one or three energy levels. They were found during our search on structure A, which has the smallest volume of all the structures. If a material system supports more than three independent energy transitions in structure A, it was discarded because as the structure increases in volume, it will support more energy levels and thus not be considered for the QD-IBSC. We propose that only the QD/barrier combinations displayed in Tab. 7.10 should be considered for the QD-IBSC.

7.4 Remarks

In this chapter, we incorporated the work from previous chapters to allow for sophisticated numerical simulations aimed at selecting more realistic material systems for the QD-IBSC. The geometry of the QD was chosen based on experimental evidence, while the size of the QD was strategically selected to bound the analysis. We included a brief discussion of kp theory in the context of the Luttinger - Kohn Hamiltonian and Pikus-Bir strain interaction for the valence bands. After the Hamiltonian is diagonalized, we observe the shift in the valence band edge is described by three deformation potential constants, unlike the shift in conduction band edge caused by a single deformation potential constant. The two deformation constants, b and d, split the heavy/light hole degeneracy with the heavy hole sitting energetically higher. Our model assumed that the shift in valence band caused by strain is determined by the Eq. 7.41.

We then proceeded to describe the model that would be used to find potential material systems

Table 7.10: An exhaustive list of potential QD-IBSC material systems that meet the design criteria for various molar concentrations.

Barrier material	QD material	Barrier material	QD material
AlGaAs	InAsN	AlAsSb	InAsN
AlGaAs	InPN	AlAsSb	InAsSb
AlInAs	InPN	AlAsSb	InPSb
AlInAs	InAsN	GaAsSb	InAsN
AlInAs	InPSb	GaAsSb	InPN
GaInP	InPN	AlAsP	InAsN
AlInP	InAsN	GaPSb	InPN
AlGaP	InAsN	AlPSb	InAsN
AlInSb	InASb	AlPSb	InAsSb
AlInSb	InPSb	AlPSb	InPSb
AlGaSb	InAsSb	GaPN	InAsN
AlAsSb	InPN	GaPN	InPN

for the QD-IBSC. We outlined the design criteria, which potential materials had to meet in order to be considered. The QD/barrier material system must have a lattice mismatch, valence band offset must be small, the large energy gap E_g must at least be 1.70 eV, and the system would allow for the maximum uncertainty. We then applied our theoretical model to InAs QDs embedded in GaAs and found it in agreement with other theoretical studies and experimental results.

Using our proposed model, we performed a thorough search of materials that have energetic optimized levels to achieve theoretical efficiencies for systems that support one and two intermediate bands under unconcentrated and fully concentrated light. Our search included all possible QD and barrier permutations on the six different structures. From the results, we propose that these material systems are potential QD-IBSC materials. In essence, we have performed a study that narrows down the likely candidate materials for this novel device. These materials deserve additional research attention both experimentally and theoretically.

Chapter 8: Final Remarks and Outlook

In this thesis, we have proposed material systems that are considered candidates for the novel QD-IBSC device aimed at increasing the efficiency beyond the thermodynamic limits of the convention solar conversion device. Our study is a first attempt to narrow down the potential material systems by performing numerical experiments based on realistic assumptions. Although previous attempts at the QD-IBSC (see [104, 105, 106]) have successfully demonstrated the basic operating principals of the IBSC, they have failed to capture the increased efficiency over conventional devices because the materials failed to satisfy one or more of the design criteria in our model. To this end, our research provides a critical step in a breakthrough of photovoltaic conversion efficiency.

While the chapters built upon each other in logical succession, we did dedicate an entire chapter to the development of the finite element method in the context of heterostructures as a means to model the QD-IBSC. Although there is a wealth of literature on the method and its development, we found its applications in physics lacking. This is the reason why we felt it necessary for Chap. 4 and 5, the detailed description of the MATLAB program ultimately used in our model. It is our belief that the method is underutilized in physics, as demonstrated by the lack of literature in the field, and could be further developed to enhance our understanding of, for example, quantum mechanics. This presents an opportunity to mature the method beyond our initial treatment. It would be useful to develop a magnetic matrix when magnetic fields are present in the problem, to develop the position and momentum operators, and develop how matrix operations are carried out. In addition, applying FEM to interesting quantum mechanical problems that have been considered intractable would obviously be beneficial.

We justified using a less sophisticated Hamiltonian by only including physics that was of real significance due to the number of possible material permutations. However, now that we have identified potential QD-IBSC materials, we could further refine that pool of materials by employing a more sophisticated Hamiltonian, i.e. coupling the valence to the conduction band, including piezoelectric effect and Coulomb interaction, as computational efficiency becomes less important. In addition to a more sophisticated Hamiltonian, it would be appropriate to refine the geometry of a specific material system if there is previous work that says otherwise.

The aspiration of our work is to create a material breakthrough that will spur additional research on the QD-IBSC design attempting to mature the technology. The QD-IBSC is a novel but very new device that will not only need the right material selection but must operate based on the IBSC operating principles. Therefore, additional research on how our proposed material systems closely match the IBSC operating principles is needed. However, the basic ideas for designing devices are not much different than from those described in this thesis.

Bibliography

[1] I. Vurgaftman, J.R. Meyer, and L.R. Ram-Mohan. Band parameters for III-V compound semiconductors and their alloys. *Journal of Applied Physics*, 89:5816–5875, 2001.

[2] Office of Integrated Analysis and Forecasting. International energy outlook 2010. Technical report, U.S. Energy Information Administration, 2010.

[3] M. K. Hubbert. Nuclear energy and the fossil fuels. *Drilling and Production Practice*, 1956.

[4] Energy Efficiency and Renewable Energy. 2008 solar technologies market report. Technical report, U.S. Department of Energy, 2010.

[5] Office of Integrated Analysis and Forecasting. Annual energy outlook 2011. Technical report, U.S. Energy Information Administration, 2011.

[6] Albert Einstein. On a heuristic viewpoint concerning the production and transformation of light. *Annalen der Physik*, 17:132–148, 1905.

[7] W. Shockley and H.J. Queisser. Detailed balance limit of efficiency of p-n junction solar cells. *Journal of Applied Physics*, 32(3), 1961.

[8] R.A. Serway, C. J. Moses, and C. A. Moyer. *Modern Physics*. Thomson Learning, Singapore, 1997.

[9] Jenny Nelson. *The Physics of Solar Cells*. Imperial College Press, London, 2003.

[10] C. Kittel. *Introduction to Solid State Physics*. John Wiley and Sons, Hoboken, 2005.

[11] A. Luque and A. Marti. Entropy production in photovoltiac conversion. *Physical review B*, 55:6994–6999, 1997.

[12] P.T. Landsberg. *Recombination in Semiconductors*. Cambridge University Press, New York, 1991.

[13] M. Shur. *Physics of Semiconductor Devices*. Prentice Hall, Englewood Cliffs, 1990.

[14] M. S. Tyagi. *Introduction to Semiconductor Materials and Devices*. Wiley, Chichester, 1991.

[15] L.E. Ballentine. *Quantum Mechanics: A Modern Development*. World Scientific, Singapore, 1998.

[16] A. Luque and A. Marti. Increasing the efficiency of ideal solar cells by photon induced transitions at intermediate levels. *Physical Review Letters*, 78(26), 1997.

[17] R. K. Pathria. *Statistical Mechanics Second Edition*. Elsevier, Burlington, MA, 2005.

[18] G. L. Araujo and A. Marti. Absolute limiting efficiencies for photovoltiac energy conversion. *Solar Energy Materials and Solar Cells*, 33:213–240, 1994.

[19] Martin A. Green. *Third Generation Photovoltaics*. Springer, New York, 2006.

[20] J. McDougall and E. C. Stoner. The computation of fermi-dirac functions. *Phil. Trans. Roy. Soc. Lond.*, A.237(67), 1938.

[21] J. S. Blakemore. Approximations for fermi-dirac integrals, especially the function $F_{1/2}(\eta)$ used to describe electron density in a semiconductor. *Solid-State Electronics*, 25(11), 1982.

[22] E. W . Ng, C. J. Devine, and R. F. Tooper. Chebyshev polynomial expansion of bose-einstein functions of orders 1 to 10. *Mathematics of Computation*, 23(107), 1969.

[23] R. W. Hamming. *Numerical Methods for Scientists and Engineers*. Dover, New York, 1962.

[24] J. P. Colinge and C. A. Colinge. *Physics of Semiconductor Devices*. Springer, New York, 2006.

[25] P. Matagne and J.P. Leburton. Quantum dots: Artificial atoms and molecules. In S. Bandyopadhyay and H.S. Nalwa, editors, *Quantum Dots and Nanowires*. American Scientific, Stevenson Ranch, 2003.

[26] N. G. Anderson. On quantum well solar cell efficiencies. *Physica E*, 14:126–131, 2002.

[27] A. Luque and A. Marti. Thermodynamic consistency of sub-bandgap absorbing solar cell proposals. *IEEE Transactions on Electron Devices*, 49:2118–2124, 2001.

[28] A. Marti, L. Cuandra, and A Luque. Quantum dot intermediate band solar cell. *28th IEEE PVSC*, pages 940–943, 2000.

[29] A. Luque and A. Marti. A metallic intermediate band high efficiency solar cell. *Progress In Photovoltaics: Research and Applications*, 9:73–86, 2001.

[30] R. Gilmore. *Elementary Quantum Mechanics In One Dimension*. John Hopkins, Baltimore, 2004.

[31] J. Tersoff, C. Teichert, and M. G. Lagally. Self-organization in growth of quantum dot superlattices. *Physical Review Letters*, 76:1675–1678, 1996.

[32] D. Leonard, M. Krishnamurthy, C. M. Reaves, S. P. Denbaars, and P. M. Petroff. Direct formation of quantum-sized dots from uniform coherent islands of ingaas on gaas surfaces. *Applied Physics Letters*, 63:3203–3205, 1993.

[33] R. Notzel, J. Temmyo, and T. Tamamura. Self-organized growth of strained ingaas quantum disks. *Nature*, 369:131–133, 2011.

[34] K. Nishi, T. Anan, A. Gomyo, S. Kohmoto, and S. Sugou. Spontaneous lateral alignment of in 0.25ga 0.75as self-assembled quantum dots on (311)b gaas grown by gas source molecular beam epitaxy. *Applied Physics Letters*, 70:3579–3581, 1997.

[35] K. Sears, S. Mokkapati, H. H. Tan, and C. Jagadish. *Self-Assembled Quantum Dots*. Springer, 2008.

[36] G. Bastard. Superlattice band structure in the envelope-function approximation. *Physical Review B*, 24:5693–5697, 1981.

[37] J. C. Slater and G. F. Koster. Simplified lcao method for the periodic potential problem. *Physical Review*, 94:1498–1524, 1954.

[38] M. Levy, C. Honsberg, A. Marti, and A. Luque. Quantum dot intermediate band solar cell material systems with negligible valence band offsets. *31st IEEE PVSC*, 2005.

[39] A. Marti, L. Cuadra, and A. Luque. Partial filling of a quantum dot intermediate band for solar cells. *IEEE Transactions on Electron Devices*, 48:2394 – 2399, 2001.

[40] P. Wurfel. *Physics of Solar Cells: From Basic Principles to Advanced Concepts*. Wiley, Weinheim, 2009.

[41] E. Schödinger. Quantization as an eigenvalue problem (part i). *Annalen der Physik*, 79, 1926.

[42] M. J. Turner, R. W. Clough, H. C. Martin, and L. J. Topp. Stiffness and deflection analysis of complex structures. *Journal of Aeronautical Sciences*, 23:805–843, 1956.

[43] O. C. Zienkiewicz and Y. K. Cheung. Finite elements in the solution of field. *The Engineer*, 220:507–510, 1965.

[44] W. Yourgrau and S. Mandelstam. *Variational Principles in Dynamics and Quantum Theory.* Dover, New York, 1968.

[45] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu. *The finite element method: Its basis and fundamentals.* Elsevier, Burlington, MA, 2005.

[46] L. Ramdas Ram-Mohan. *Finite Element and Boundary Element Applications in Quantum Mechanics.* Oxford University Press, New York, 2002.

[47] K. J. Bathe and E. L. Wilson. *Numerical Methods in Finite Element Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1976.

[48] Y. W. Kwon and H. Bang. *The finite element method using MATLAB.* CRC Press, Boca Raton, FL, 2000.

[49] G. Strang and G. Fix. *An Analysis of the Finitie Element Method.* Wellesley-Cambridge Press, Wellesley, MA, 2 edition, 2008.

[50] P.-O. Persson and G. Strang. A simple mesh generator in matlab. *SIAM Review*, 46:329–345, 2004.

[51] Waterloo Maple Inc. Maple, 2009. version 13.0.

[52] The Mathworks inc. Matlab, 2009. version 7.9.0.529 (R2009b).

[53] Simulia. Abaqus. http://www.simulia.com/.

[54] PDE Solutions Inc. Flexpde 6. http://www.pdesolutions.com/.

[55] ANSYS Inc. Ansys. http://www.ansys.com/.

[56] Innovative Numerical Technologies. Diffpack. http://www.diffpack.com/.

[57] B. Delaunay. Sur la sphere vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, 7:793–800, 1934.

[58] John D'Errico. Inhull, 2006.

[59] D. Zwillinger (ed.). *Standard Mathematical Tables and Formulae.* CRC Press, Boca Raton, 30th edition, 1996.

[60] S. Jenks and R. Gilmore. Quantum dot solar cell: Materials that produce two intermediate bands. *Journal of Renewable and Sustainable Energy*, 2(1):013111, 2010.

[61] J. Bardeen and W. Shockley. Deformation potentials and mobilities in non-polar crystals. *Physical Review*, 80:72–80, 1950.

[62] C. Herring and E. Vogt. Transport and deformation-potential theory for many-valley semiconductors with anisotropic scattering. *Physical Review*, 101:944–961, 1956.

[63] L. Malvern. *Introduction to the Mechanics of a Continuous Medium.* Prentice Hall, Englewood Cliffs, NJ, 1969.

[64] W. Lai, D. Rubin, and E. Krempl. *Introduction to Continuum Mechanics.* Elsevier, Amsterdam, 1996.

[65] F. Boxberg and J. Tulkki. Theory of the electronic structure and carrier dynamics of strain-induced (ga, in)as quantum dots. *Reports on Progress in Physics*, 70:1425–1471, 2007.

[66] S. Timoshenko and J. Goodier. *Theory of Elasticity, 3rd ed.* McGraw-Hill, New York, 1970.

[67] D. Logan. *A First Course in the Finite Element Method.* PWS Publishers, Boston, 1986.

[68] E. Dill. *Continuum Mechanics.* CRC Press, 2006.

[69] W. Slaughter. *The Linearized Theory of Elasticity.* Birkhauser Boston, New York, NY, 2002.

[70] L. Freund. The mechanics of electronic materials. *International Journal of Solids and Structures*, 37:185–196, 2000.

[71] S. Chang. *The Physics of Photonic Devices.* Wiley, 2009.

[72] J. Eshelby. Distortion of a crystal by point imperfections. *Journal of Applied Physics*, 25:255, 1954.

[73] J. Eshelby. The continuum theory of lattice defects. *Solid State Physics*, 3:79, 1956.

[74] M. Kuo, T. Lin, K. Hong, B. Liao, H. Lee, and C. Yu. Two step strain analysis of self assembled inas/gaas quantum dots. *Semiconductor Science and Technology*, 21:626–632, 2006.

[75] A. Andreev, J. Downes, D. Faux, and E. O'Reilly. Strain distributions in quantum dots of arbitrary shape. *Journal of Applied Physics*, 86:297–305, 1999.

[76] G. Bir and G. Pikus. *Symmetry and Strain Induced Effects in Semiconductors.* Wiley, New York, 1974.

[77] C. Hamaguchi. *Basic Semiconductor Physics.* Springer, Heidelberg, 2010.

[78] M. Grundmann, O. Stier, and D. Bimberg. Inas/gaas pyramidal quantum dots: Strain distrbution, optical phonons, and electronic structure. *Physical Review B*, 52:11969–11981, 1995.

[79] M. Yang, J. Sturm, and J. Prevost. Calculation of band alignments and quantum confinement effects in zero- and one-dimensional pseudomorphic structures. *Physical Review B*, 56:1973–1980, 1997.

[80] M.S. Miller, J.O. Malm, M. E. Pistol, S. Jeppesen, B. Kowalski, K. Georgsson, and L. Samuelson. Stacking inas islands and gaas layers: Strongly modulated one-dimensional electronic systems. *Journal of Applied Physics*, 80:3360–3365, 1996.

[81] G. S. Solomon, J. A. Trezza, J. A. Marshall, and J. A. Harris. Vertically aligned and electronically coupled growth induced inas islands in gaas. *Physical Review Letters*, 76:952–955, 1996.

[82] Q. Xie, A. Madhukar, P. Chen, and N. P. Kobayashi. Vertically self-organized inas quantum box islands on gaas (100). *Physical Review Letters*, 75:25422545, 1995.

[83] M. Moreno, A. Trampert, B. Jenichen, L. Daweritz, and K. H. Ploog. Correlation of structure and magnetism in gaas with embedded mn(ga)as magnetic nanoclusters. *Journal of Applied Physics*, 92:4672–4678, 2002.

[84] A. Lenz, R. Timm, H. Eisele, Ch. Hennig, S. K. Becker, R. L. Sellin, U. W. Pohl, D. Bimberg, and M. Dahn. Reversed truncated cone composition distribution of in 0.8 ga 0.2 as quantum dots overgrown by an in 0.1 ga 0.9 as layer in a gaas matrix. *Applied Physics Letters*, 81:5150–5153, 2002.

[85] S. Ruvimov, P. Wemer, K. Scheerschmidt, U. Gbsele, J. Heydenreich, U. Richter, N. N. Ledentsov, M. Grundmann, D. Bimberg, V. M. Ustinov, A. Y. Egorov, P. S. Kop'ev, and Z. I. Alferov. Structural characterization of (in,ga)as quantum dots in a gaas matrix. *Physical Review B*, 51:1476614769, 1995.

[86] B. R. Bennett, R. Magno, and B. V. Shanabrook. Molecular beam epitaxial growth of insb, gasb, and alsb nanometerscale dots on gaas. *Applied Physics Letters*, 68:505–507, 1996.

[87] F. Heinrichsdorff, A. Krost, M. Grundmann, D. Bimberg, A. Kosogov, and P. Werner. Self-organization processes of ingaas/gaas quantum dots grown by metalorganic chemical vapor deposition. *Applied Physics Letters*, 68:3284–3286, 1996.

[88] A. Ponchet, A. L. Corre, H. L. Haridon, B. Lambert, and S. Salaun. Relationship between self-organization and size of inas islands on inp(001) grown by gas-source molecular beam epitaxy. *Applied Physics Letters*, 67:1850, 1995.

[89] V. Zwiller, M. E. Pistol, M. A. Odnoblyudovtt, and L. Samuelson. Temperature studies of single inp quantum dots. *7th Int. Symp. "Nanostructures: Physics and Technology"*, pages 28–30, 1999.

[90] J. M. Luttinger and W. Kohn. Motion of electrons and holes in perturbed periodic fields. *Physical Review*, 97:869–883, 1954.

[91] L. C. Lew Yan Voon and M. Willatzen. *The k p Method, Electronic Properties of Semiconductors*. Springer, 2009.

[92] D. A. Broido and L. J. Sham. Effective masses of hales at gaas-algaas heterojunctions. *Physical Review B*, 31:888–892, 1985.

[93] E. O. Kane. Energy band structure in p-type germanium and silicon. *Journal of Physical Chemical Solids*, 1:82–99, 1956.

[94] G. .E. Pikus and G. L. Bir. Effect of deformation on the hole energy spectrum of germanium and silicon. *Soviet Physics Solid State*, 1:1502, 1960.

[95] S. L. Chuang. *Physics of Optoelectronic Devices*. Wiley, New York, 1995.

[96] T. B. Bahder. Analytic dispersion relations near the gamma point in strained zincblende crystals. *Physical Review B*, 45:1629–1637, 1992.

[97] M. Holm and M. Pistol. Calculations of the electronic structure of strained inas quantum dots in inp. *Journal of Applied Physics*, 92:932–936, 2002.

[98] C. Pryor. Eight-band calculations of strained inas/gaas quantum dots compared with one-, four-, and six-band approximations. *Physical Review B*, 57:7190–7195, 1998.

[99] Y. P. Varshni. Temperature dependence of the energy gap in semiconductors. *Physica*, 34:149, 1967.

[100] C. Bosio, J. L. Staehli, M. Guzzi, G. Burri, and R. A. Logan. Direct-energy-gap dependence on al concentration in algaas. *Physical Review B*, 38:3263–3268, 1988.

[101] G. Bais, A. Cristofoli, F. Jabeen, M. Piccin, E. Carlino, S. Rubini, F. Martelli, and A. Franciosi. Inasn/gaasn quantum-dot and inganas/gaas quantum-well emitters: A comparison. *Applied Physics Letters*, 86:233107, 2005.

[102] V. M. Daniltsev, M. N. Drozdov, Yu N Drozdov, D. M. Gaponova, O. I. Khrykin, A. V. Murel, V. I. Shashkin, and N. V. Vostokov. Ingaasn/gaas qd and qw structures grown by movpe. *Journal of Crystal Growth*, 248:343–347, 2003.

[103] O. Schumann, L. Geelhaar, H. Riechert, H. Cerva, and G. Abstreiter. Morphology and optical properties of inasn quantum dots. *Journal of Applied Physics*, 96:2832–2840, 2004.

[104] V. Popescu, G. Bester, M. C. Hanna, A. G. Norman, and A. Zunger. Theoretical and experimental examination of the intermediate-band concept for strain-balanced (in,ga)as/ga(as,p) quantum dot solar cells. *Physical Review B*, 78:205321, 2008.

[105] A. Luque, A. Marti, C. Stanley, N. Lopez, L. Cuandra, D. Zhou, J. L. Pearson, and A. McKee. General equivalent circuit for intermediate band devices: Potentials, currents and electroluminescence. *Journal of Applied Physics*, 96:903–909, 2004.

[106] S. Suraprapapich, S. Thainoi, S. Kanjanachuchai, and S. Panyakeow. Quantum dot integration in heterostructure solar cell. *Journal of Solar Energy Materials and Solar Cells*, 90:2968–2974, 2006.

Appendix A: Schödinger's Equation in Variational Form

Here we develop Schödinger's equation from the Hamilton-Jacobian formulation of classical mechanics, just as Schödinger did himself. We start with the Hamilton-Jacobian equation

$$H\left(x, \frac{\partial S}{\partial x}\right) = E \tag{A.1}$$

and define the "wavefunction" as

$$\psi(x) = e^{\frac{iS}{\hbar}} \tag{A.2}$$

$$S(x) = -i\hbar \ln \psi(x) \tag{A.3}$$

such that the Hamiltonian is

$$H\left(x, \frac{\partial S}{\partial x}\right) = \frac{1}{2m}\left(\frac{\partial S}{\partial x}\right)^2 + V \tag{A.4}$$

$$= \frac{1}{2m}\left(\frac{-i\hbar}{\psi(x)}\frac{\partial \psi(x)}{\partial x}\right)^2 + V \tag{A.5}$$

Further, we can define the Lagrangian, L, as $\frac{\partial S}{\partial x}\dot{x} - H$ such that

$$L = \frac{1}{m}\left(\frac{\partial S}{\partial x}^2\right) - \frac{1}{2m}\left(\frac{\partial S}{\partial x}\right)^2 - V \tag{A.6}$$

$$= \frac{1}{2m}\left(\frac{-i\hbar}{\psi(x)}\frac{\partial \psi(x)}{\partial x}\right)^2 - V \tag{A.7}$$

$$= \frac{-\hbar^2}{2m}\left(\frac{1}{\psi(x)}\right)^2\left(\frac{\partial \psi(x)}{\partial x}\right)^2 - V \tag{A.8}$$

The Lagrangian is used to construct the action, $\int L dx$.

$$A = \int \frac{-\hbar^2}{2m}\left(\frac{1}{\psi(x)}\right)^2\left(\frac{\partial \psi(x)}{\partial x}\right)^2 - V\, dx \tag{A.9}$$

subject to the constraint

$$\int \psi^*(x)\psi(x)dx = 1 \tag{A.10}$$

We can consider $\psi^*(x)$ and $\psi(x)$ as two independent fields and place a variation with respect to $\psi^*(x)$ such that the integral is stationary.

$$\delta A = \delta \int \frac{-\hbar^2}{2m}\left(\frac{1}{\psi(x)}\right)^2\left(\frac{\partial \psi(x)}{\partial x}\right)^2 - V dx = 0$$

$$\delta A = \delta \int \frac{\hbar^2}{2m}\left(\frac{\partial \psi^*(x)}{\partial x}\frac{\partial \psi(x)}{\partial x}\right) + \psi^*(x)V\psi(x)dx = 0 \tag{A.11}$$

When we include the Lagrange multiplier, E, to Eq. A.11, the variational form of Schödinger's equation is

$$\delta \int \frac{\hbar^2}{2m} \left(\frac{\partial \psi^*(x)}{\partial x} \frac{\partial \psi(x)}{\partial x} \right) + \psi^*(x) V \psi(x) - \psi^*(x) E \psi(x) dx = 0 \tag{A.12}$$

where the Lagrange multiplier can be interpreted as energy.

Appendix B: FEM Source Code

This appendix contains the entire source code for the FEM program used to make the calculations referred to in the text. The main function from each of the FEM stages is displayed first followed by the subroutine functions.

B.1 2D Preprocessing Code

```
function [p,t,vert] = QWire_Mesh( QW_fd,B_fd,fh,h0,QW_bbox,B_bbox )
%Quantum wire mesh generator using Distmesh
%   This generator is specifically taylored to solve for the
%   the energy eigenvalues and wavefunctions of a quantum wire
%   embedded in barrier material using FEM.
%
%   [P,T,VERT]=QWIRE_MESH(QW_FD,B_FD,FH,H0,QW_BBOX,B_BBOX)
%      P:         Node positions (Nx2)
%      T:         Triangle indices (NTx3)
%      VERT:      Nodes on the quantum wire surface
%      QW_FD:     Quantum Well Distance function d(x,y)
%      B_FD:      Barrier Distance function d(x,y)
%      FH:        Scaled edge length function h(x,y)
%      H0:        Initial edge length
%      QW_BBOX:   Quantum Well material bounding box [xmin,ymin; xmax,ymax]
%      B_BBOX:    Barrier material bounding box [xmin,ymin; xmax,ymax]
%---------------------------------------------------------------------------
% Create a mesh for the quantum wire material using distmesh2D
[QW_coord,QW_nodes]=distmesh2d(QW_fd,fh, h0,QW_bbox,[]);
[QW_coord,QW_nodes]=fixmesh(QW_coord,QW_nodes);
QW_surface=unique(boundedges(QW_coord,QW_nodes));
fixed=[QW_coord(QW_surface(:),1),QW_coord(QW_surface(:),2)];
% Create a mesh for the barrier material using distmesh 2D
%
[B_coord,B_nodes]=distmesh2d(B_fd,fh,h0,B_bbox,fixed);
[B_coord,B_nodes]=fixmesh(B_coord,B_nodes);
B_surface=unique(boundedges(B_coord,B_nodes));
% Convert coordinates to desired significant digits
%
B_coord=round(B_coord*1000)/1000;
QW_coord=round(QW_coord*1000)/1000;
fixed=round(fixed*1000)/1000;
vert = fixed;
%
% 1. Get coordinates of all the nodes on edges of the barrier material
edges=[B_coord(B_surface(:),1),B_coord(B_surface(:),2)];
%
% 2. Start to systematically eliminate nodes associated with outer edges
for i=1:2
    for j=1:2
        [r c]=find(edges == B_bbox(i,j));
        edges(r,:)=[];
    end
end
%
% 3. Start to systematically eliminate nodes associated with nodes on
```

```
% the QW edges.
[index]=ismember(edges,fixed,'rows');
edges(index,:)=[];
%
% 4. Eliminate "orphan" nodes
[index]=ismember(B_coord,edges,'rows');
B_coord(index,:)=[];
%
% 5. Complete the final mesh
p=unique([B_coord;QW_coord],'rows');
t=delaunayn(p);
%
% 6. View the final mesh
trimesh(t,p(:,1),p(:,2),zeros(size(p,1),1))
view(2),axis equal,axis off,drawnow
end
```

B.1.1 Example of a relative edge length function

```
function h = fh_finite_2D(p)
%Fh_finite_2D variable edge length that uses a distance function
%   This function is used with QWire_Mesh when it is desired to have more
%   elements near the quantum and barrier material interface.  It uses the
%   distance function of the quantum wire.
%
%   [H]=FH_FINITE_2D(P)
%      H:   EDGE LENGTH
%      P:   NODE POSITION (Nx2)
%-------------------------------------------------------------------------
%   Variables:
%       d1: The distance function of the quantum wire.
%       h1: The calibration used in defining the edge lengths, linear
%       function.
% h: Minimum distance, either h1 or scalar.
d1=dcircle(p,0,0,20);
h1=5+0.2*(abs(d1));
h=min(h1,10);
end
```

B.2 3D Preprocessing Code

```
function [p,t,vert] = QDot_Mesh( QD_fd,B_fd,fh,h0,QD_bbox,B_bbox )
%Quantum dot mesh generator using Distmesh
%   This generator is specifically tailored to solve for the
%   the energy eigenvalues and wavefunctions of a quantum dot
%   embedded in barrier material using FEM.
%
%   [P,T,VERT]=QWIRE_MESH(QW_FD,FH,H0,BBOX,PFIX)
%      P:         Node positions (Nx2)
%      T:         Tetrahedral indices (NTx3)
%      VERT:      Verticies of Quantum Dot
%      QD_FD:     Quantum Dot Distance function d(x,y,z)
%      B_FD:      Barrier Distance function d(x,y,z)
%      FH:        Scaled edge length function h(x,y,z)
%      H0:        Initial edge length
%      QD_BBOX:   Quantum Dot material bounding box [xmin,ymin,zmin;
%      xmax,ymax,zmax]
%      B_BBOX:    Barrier material bounding box [xmin,ymin,zmin;
```

```
%      xmax,ymax,zmax]
%-------------------------------------------------------------------------
% Create a mesh for the quantum dot material using distmesh ND
[QD_coord,QD_nodes]=distmeshnd(QD_fd,fh, h0,QD_bbox,[]);
[QD_coord,QD_nodes]=fixmesh(QD_coord,QD_nodes);
QD_surface=unique(surftri(QD_coord,QD_nodes));
fixed=[QD_coord(QD_surface(:),1),QD_coord(QD_surface(:),2),QD_coord(QD_surface(:),3)];
% Create a mesh for the barrier material using distmesh ND
%
[B_coord,B_nodes]=distmeshnd(B_fd,fh,h0,B_bbox,fixed);
[B_coord,B_nodes]=fixmesh(B_coord,B_nodes);
B_surface=unique(surftri(B_coord,B_nodes));
% Convert coordinates to desired significant digits
%
B_coord=round(B_coord*1000)/1000;
QD_coord=round(QD_coord*1000)/1000;
fixed=round(fixed*1000)/1000;
vert = fixed;
% 1. Get coordinates of all the nodes on edges of the barrier material
edges=[B_coord(B_surface(:),1),B_coord(B_surface(:),2),B_coord(B_surface(:),3)];
% 2. Start to systematically eliminate nodes on barrier bounding cube
for i=1:2
    for j=1:3
        [r c]=find(edges == B_bbox(i,j));
        edges(r,:)=[];
    end
end
% 3. Start to systematically eliminate nodes associated with nodes on
% the QD surface.
%
[index]=ismember(edges,fixed,'rows');
if (~isempty(index))
edges(index,:)=[];
end
% 4. Eliminate "orphan" nodes
%
[index]=ismember(B_coord,edges,'rows');
B_coord(index,:)=[];
% 5. Complete the final mesh
%
p=unique([B_coord;QD_coord],'rows');
t=delaunayn(p);
%
% 6. View mesh
simpplot(p,t,'p(:,2)>0');
drawnow
end
```

B.2.1 Five Sided Pyramid Distance Function

```
function d_signed=dFiveSidedPyramid(p,pv)
%dFiveSidedPyramid generates the signed distance from a set up points
%to a five sided pyramid.  This function is used as a distance function
%in DistMesh.
%
%   DFIVESIDEDPYRAMID(P,PV)
%      D_SIGNED:     SIGNED DISTANCE TO FIVE SIDED PYRAMID SURFACE (Nx1)
%      P:            POINTS (NX3)
%      PV:           VERTEX COORDINATES OF 5-SIDED PYRAMID
```

```
%
%-------------------------------------------------------------------------
%   In order to determine the signed distance, we first decompose
%   the surfaces of the polyhedron into surface triangles.  This is
%   accomplished by composing the surface triangles in a NX3 matrix, with N
%   number of surface triangles each containing its 3 vertices.
%
%   Variables:
%       T: MATRIX REPRESENTING THE SURFACE TRIANGLES (6X3)
%       NT: NUMBER OF SURFACE TRIANGLES
%       TP: VERTEX POINTS OF SURFACE TRIANGLE (3X3)
%       DS: DISTANCE OF POINTS FROM SURFACE TRIANGLE USING FUNCTION
%       D: UNSIGNED DISTANCE TO FIVE SIDED PYRAMID SURFACE (NX1)
%   Functions:
%       POLYGON_CENTROID_3D - calculates the centroid of the polyhedron.
%       TRIANGLE_POINT_DIST_3D - generates the distance from a set of
%           points to a triangle in 3D.
%       INHULL - tests if a set of points are inside a convex hull.
%
T=[1 2 5; 2 3 5; 3 4 5;4 5 1;1 2 4; 2 3 4];
centroid = polygon_centroid_3d(T,pv);
%
%   The distance from each surface triangle is computed and then minimized
%   to determine the distance.
nT=size(T,1);
%
for j=1:nT
    TP = [pv(T(j,1),:);pv(T(j,2),:);pv(T(j,3),:)];
    ds(:,j) = triangle_point_dist_3d(TP,p,centroid);
end

d=min(ds,[],2);
%   The function inhull is used to determine the signed distance.  If the
%   point lies within the polyhedron, the distance is negative.  If
%   the point lies outside the polyhedron, the distance is positive.
d_signed=(-1).^(inhull(p,pv)).*d;
end
```

B.2.2 Rectangular Prism Distance Function

```
function d_signed=drectangle_3D(p,pv)
%drectangle_3D generates the signed distance from a set up points
%to a rectangular prism.  This function is used as a distance function
%in DistMesh.
%
%   DRECTANGLE_3D(P,PV)
%      D_SIGNED:     SIGNED DISTANCE TO RECTANGULAR PRISM SURFACE (Nx1)
%      P:            POINTS (NX3)
%      PV:           VERTEX COORDINATES OF RECTANGLULAR PRISM (8X3)
%
%-------------------------------------------------------------------------
%   In order to determine the signed distance, we first decompose
%   the surfaces of the polyhedron into surface triangles.  This is
%   accomplished by composing surface triangles in a NX3 matrix, with N
%   number of surface triangles each containing its 3 vertices.
%
%   Variables:
%       T: MATRIX REPRESENTING THE SURFACE TRIANGLES (12X3)
%       NT: NUMBER OF SURFACE TRIANGLES
```

```
%       TP: VERTEX POINTS OF SURFACE TRIANGLE (3X3)
%       DS: DISTANCE OF POINTS FROM SURFACE TRIANGLE USING FUNCTION
%       D: UNSIGNED DISTANCE TO RECTANGULAR PRISM SURFACE (NX1)
%   Functions:
%       POLYGON_CENTROID_3D - calculates the centroid of the polyhedron.
%       TRIANGLE_POINT_DIST_3D - generates the distance from a set of
%           points to a triangle in 3D.
%       INHULL - tests if a set of points are inside a convex hull.
%
T=[1 2 6;1 6 5;2 3 7;2 7 6;3 4 8;3 8 7;1 4 8;1 8 5;1 2 3;1 4 3;5 6 7;5 8 7];
centroid = polygon_centroid_3d(T,pv);
%
%   The distance from each surface triangle is computed and then minimized
%   to determine the distance.
nT=size(T,1);
%
for j=1:nT
    TP = [pv(T(j,1),:);pv(T(j,2),:);pv(T(j,3),:)];
    ds(:,j) = triangle_point_dist_3d(TP,p,centroid);
end

d=min(ds,[],2);
%   The function inhull is used to determine the signed distance.  If the
%   point lies within the polyhedron, the distance is negative.  If
%   the point lies outside the polyhedron, the distance is positive.
d_signed=(-1).^(inhull(p,pv)).*d;
end
```

B.2.3 Polygon_Centroid_3D

```
function centroid = polygon_centroid_3d ( t, v )
%Polygon_centroid_3d calculates the centroid of the polyhedron.
%
%   POLYGON_CENTROID_3D ( T, V )
%       CENTROID: CENTROID OF POLYHEDRON
%       T: MATRIX REPRESENTING SURFACE TRIANGLES OF POLYHEDRON
%       V: VERTICIES OF POLYHEDRON
%--------------------------------------------------------------------------
area = 0.0;
centroid(1:3) = 0.0;
for i = 1:size(t,1)
tri = [ v(t(i,1),:); v(t(i,2),:); v(t(i,3),:) ];
    c = cross(tri(2,:)-tri(1,:),tri(3,:)-tri(1,:));
    area_triangle = 0.5 * sqrt ( sum ( c.^2 ) );
    area = area + area_triangle;
    centroid = centroid + area_triangle ...
    * ( v(t(i,1),:) + v(t(i,2),:) + v(t(i,3),:) ) / 3.0;

end
if ( area == 0.0 )
centroid = v(t(1,1),:);
else
    centroid = centroid / area;
end
return
end
```

B.2.4 Triangle_point_dist_3d

```
function dist = triangle_point_dist_3d ( t, p, centroid )
%Triangle_point_dist_3d generates the distance from a set of points
%to a triangle in 3D.
%
%   TRIANGLE_POINT_DIST_3D(T,P)
%       DIST: DISTANCE FROM TRIANGLE (Nx1)
%       T: TRIANGLE VERTICIES (3X3)
%       P: POINTS (NX3)
%       CENTROID: CENTROID OF POLYHEDRON
%
%--------------------------------------------------------------------------
%   This function is used to find the distance of a set of 3D points to a
%   triangle defined by its vertices.
%
%   The algorithm does the following:
%       1.  Determine the distances to each triangle edge and vertex.
%       These distances are minimized.
%       2.  Determine the distances to the plane the triangle lies on.
%       3.  Project the points to the plane on the triangles plane.
%       4.  Determine if the projected points are inside or outside the
%       triangle.
%       5.  For those points inside, the distance is to the plane the
%       triangle lies on.
%
%   Variables:
%       DIST2: DISTANCE FROM POINTS TO SEGMENTS OR VERTICIES IN TRIANGLE
%       DIST3: DISTANCE FROM POINTS TO PLANE IN TRIANGLE
%       P_0: POINTS PROJECTED ON PLANE OF TRIANGLE (NX3)
%       FLAG: NX1 ARRAY USED TO DISTINGUISH IF POINT LIES OUTSIDE OR INSIDE
%           OF TRIANGLE.  IF OUTSIDE, FLAG IS 0.  IF INSIDE, FLAG IS 1 (NX1)
%       IND: INDICES CORRESPONDING TO FLAG EQUAL TO 1.
%   Functions:
%       SEGMENT_POINT_DIST_3D - distance from a set of points
%           to a line segment in 3D.
%       PLANE_VERT_POINT_DIST_3D - computes the distance from a point to
%           plane using the 3 vertices of a triangle.
%       INHULL - tests if a set of points are inside a convex hull.
%
%   Compute the distances from the points to each of the sides or vertex using
%   the function seqment_point_dist_3d.  Minimize these distances.
%
  dist2 = segment_point_dist_3d ( t(1,:), t(2,:), p );
  dist = dist2;

  dist2 = segment_point_dist_3d( t(2,:), t(3,:), p );
  dist = min ( dist, dist2 );

  dist2 = segment_point_dist_3d ( t(3,:), t(1,:), p );

  dist = min ( dist, dist2 );
%   Compute the distance from the points to the plane using the function
%   plane_vert_point_dist_3d.  This function (plane_vert_point_dist_3d)
%   calls the two functions plane_vert2std_3d and plane_std_point_dist3d
%   which converts to the standard form of the plane and computes the
%   distance while producing a set of projected points respectively.
%   Use of the inhull function determines if the projected points lie
%   inside or outside the triangle.  To do this, we need to feed inhull a
```

```
%   fourth point not on the plane.  This allows inhull to determine those
%   points within the triangle (added 4th point creates a tetrahedron).
%   The 4th point can be chosen arbitrarily, so we use its centroid of the
% quantum dot shape.

  [dist3,p_0] = plane_vert_point_dist_3d ( t(1,:),t(2,:),t(3,:),p );
  dist4 = min ( dist , dist3 );
  flag=inhull(p_0,[t; centroid],[],1.0e-05);
  ind=find(flag);
% Replace distances
  dist(ind) = dist4(ind);
  return
end
```

B.2.5 Inhull

```
function in = inhull(testpts,xyz,tess,tol)
% inhull: tests if a set of points are inside a convex hull
% usage: in = inhull(testpts,xyz)
% usage: in = inhull(testpts,xyz,tess)
% usage: in = inhull(testpts,xyz,tess,tol)
%
% arguments: (input)
%  testpts - nxp array to test, n data points, in p dimensions
%       If you have many points to test, it is most efficient to
%       call this function once with the entire set.
%
%  xyz - mxp array of vertices of the convex hull, as used by
%       convhulln.
%
%  tess - tessellation (or triangulation) generated by convhulln
%       If tess is left empty or not supplied, then it will be
%       generated.
%
%  tol - (OPTIONAL) tolerance on the tests for inclusion in the
%       convex hull. You can think of tol as the distance a point
%       may possibly lie outside the hull, and still be perceived
%       as on the surface of the hull. Because of numerical slop
%       nothing can ever be done exactly here. I might guess a
%       semi-intelligent value of tol to be
%
%         tol = 1.e-13*mean(abs(xyz(:)))
%
%       In higher dimensions, the numerical issues of floating
%       point arithmetic will probably suggest a larger value
%       of tol.
%
%       DEFAULT: tol = 0
%
% arguments: (output)
%  in  - nx1 logical vector
%        in(i) == 1 --> the i'th point was inside the convex hull.
%
% Example usage: The first point should be inside, the second out
%
%  xy = randn(20,2)
%  tess = convhull(xy(:,1),xy(:,2));
%  testpoints = [ 0 0; 10 10];
%  in = inhull(testpts,xyz,tess)
```

```
%
% in =
%      1
%      0
%
% A non-zero count of the number of degenerate simplexes in the hull
% will generate a warning (in 4 or more dimensions.) This warning
% may be disabled off with the command:
%
%   warning('off','inhull:degeneracy')
%
% See also: convhull, convhulln, delaunay, delaunayn, tsearch, tsearchn
%
% Author: John D'Errico
% e-mail: woodchips@rochester.rr.com
% Release: 3.0
% Release date: 10/26/06

% get array sizes
% m points, p dimensions
p = size(xyz,2);
[n,c] = size(testpts);
if p ~= c
  error 'testpts and xyz must have the same number of columns'
end
if p < 2
  error 'Points must lie in at least a 2-d space.'
end

% was the convex hull supplied?
if (nargin<3) || isempty(tess)
  tess = convhulln(xyz);
end
[nt,c] = size(tess);
if c ~= p
  error 'tess array is incompatible with a dimension p space'
end

% was tol supplied?
if (nargin<4) || isempty(tol)
  tol = 0;
end

% build normal vectors
switch p
  case 2
    % really simple for 2-d
    nrmls = (xyz(tess(:,1),:) - xyz(tess(:,2),:)) * [0 1;-1 0];

    % Any degenerate edges?
    del = sqrt(sum(nrmls.^2,2));
    degenflag = (del<(max(del)*10*eps));
    if sum(degenflag)>0
      warning('inhull:degeneracy',[num2str(sum(degenflag)), ...
        ' degenerate edges identified in the convex hull'])

      % we need to delete those degenerate normal vectors
      nrmls(degenflag,:) = [];
      nt = size(nrmls,1);
```

```
    end
  case 3
    % use vectorized cross product for 3-d
    ab = xyz(tess(:,1),:) - xyz(tess(:,2),:);
    ac = xyz(tess(:,1),:) - xyz(tess(:,3),:);
    nrmls = cross(ab,ac,2);
    degenflag = repmat(false,nt,1);
  otherwise
    % slightly more work in higher dimensions,
    nrmls = zeros(nt,p);
    degenflag = repmat(false,nt,1);
    for i = 1:nt
      % just in case of a degeneracy
      nullsp = null(xyz(tess(i,2:end),:) - repmat(xyz(tess(i,1),:),p-1,1))';
      if size(nullsp,1)>1
        degenflag(i) = true;
        nrmls(i,:) = NaN;
      else
        nrmls(i,:) = nullsp;
      end
    end
    if sum(degenflag)>0
      warning('inhull:degeneracy',[num2str(sum(degenflag)), ...
        ' degenerate simplexes identified in the convex hull'])

      % we need to delete those degenerate normal vectors
      nrmls(degenflag,:) = [];
      nt = size(nrmls,1);
    end
end

% scale normal vectors to unit length
nrmllen = sqrt(sum(nrmls.^2,2));
nrmls = nrmls.*repmat(1./nrmllen,1,p);

% center point in the hull
center = mean(xyz,1);

% any point in the plane of each simplex in the convex hull
a = xyz(tess(~degenflag,1),:);

% ensure the normals are pointing inwards
dp = sum((repmat(center,nt,1) - a).*nrmls,2);
k = dp<0;
nrmls(k,:) = -nrmls(k,:);

% We want to test if:  dot((x - a),N) >= 0
% If so for all faces of the hull, then x is inside
% the hull. Change this to dot(x,N) >= dot(a,N)
aN = sum(nrmls.*a,2);

% test, be careful in case there are many points
in = repmat(false,n,1);

% if n is too large, we need to worry about the
% dot product grabbing huge chunks of memory.
memblock = 1e6;
blocks = max(1,floor(n/(memblock/nt)));
aNr = repmat(aN,1,length(1:blocks:n));
```

```
for i = 1:blocks
   j = i:blocks:n;
   if size(aNr,2) ~= length(j),
      aNr = repmat(aN,1,length(j));
   end
   in(j) = all((nrmls*testpts(j,:)' - aNr) >= -tol,1)';
end
```

B.2.6 Segment_Point_Dist_3D

```
function [dist] = segment_point_dist_3d ( p1, p2, p )
%Segment_point_dist_3d generates the distance from a set of points
%to a line segment in 3D.
%
%   SEGMENT_POINT_DIST_3D(P1,P2,P)
%       DIST: DISTANCE FROM SEGMENT (Nx1)
%       P1: FIRST VERTEX POINT OF LINE SEGEMENT (1X3)
%       P2: SECOND VERTEX POINT LINE SEGEMENT (1X3)
%-------------------------------------------------------------------------
%
%   Variables:
%       BOT: SUM SQUARED OF P2-P1
%       T: PARAMETER T
%       PN: NORMALIZED VECTOR ALONG LINE
  bot = sum ( ( p2 - p1 ).^2 );
  t = ( p - ones(size(p,1),1)*p1 ) * ( p2 - p1 )' ./ bot;
  t = max ( t, 0.0 );
  t = min ( t, 1.0 );
  pn = ones(size(p,1),1)*p1 + t * (p2 - p1 );
  dist = sqrt ( sum ( ( pn - p ).^2,2 ) );
  return
end
```

B.2.7 Plane_Vert_Point_Dist_3D

```
function [dist,p_0] = plane_vert_point_dist_3d ( p1, p2, p3, p )
%Plane_vert_point_dist_3d computes the distance from a point to plane using
%the 3 vertices of a triangle.  It calls the function plane_vert2std_3d to
%convert the plane to standard form;
%   STANDARD FORM IS
%   A * X + B * Y + C * Z + D = 0;
%
%Also calls the function plane_std_point_dist_3d to determine the
%distance from a set of points to a plane using the standard form's
%coefficients a,b,c, and d and gives the projected points.
%
%   PLANE_VERT_POINT_3D(P1,P2,P3,P)
%       DIST: DISTANCE BETWEEN POINT AND PLANE (NX1)
%       P_0: PROJECTED POINTS (NX3)
%       P1: FIRST POINT (TRIANGLE VERTEX)
%       P2: SECOND POINT (TRIANGLE VERTEX)
%       P3: THIRD POINT (TRIANGLE VERTEX)
%       P: POINTS (NX3)
%-------------------------------------------------------------------------
%   Variables:
%       A: COEFFICIENT A
%       B: COEFFICIENT B
%       C: COEFFICIENT C
%       D: COEFFICIENT D
```

```
%       P_0: PROJECTED POINTS (Nx3)
  [ a, b, c, d ] = plane_vert2std_3d ( p1, p2, p3 );
  [dist, p_0] = plane_std_point_dist_3d ( a, b, c, d, p );
  return
end
```

B.2.8 Plane_vert2std_3d

```
function [ a, b, c, d ] = plane_vert2std_3d ( p1, p2, p3 )
%Plane_vert2std_3d converts 3 points on a plane to the standard form in 3D.
%   STANDARD FORM IS
%   A * X + B * Y + C * Z + D = 0;
%
%   PLANE_EXP2IMP_3D(P1,P2,P3)
%       A: COEFFICIENT A
%       B: COEFFICIENT B
%       C: COEFFICIENT C
%       D: COEFFICIENT D
%       P1: FIRST POINT (TRIANGLE VERTEX)
%       P2: SECOND POINT (TRIANGLE VERTEX)
%       P3: THIRD POINT (TRIANGLE VERTEX)
%--------------------------------------------------------------------------
  a = (p2(2) - p1(2))*(p3(3) - p1(3)) - (p2(3) - p1(3))*(p3(2) - p1(2));
  b = (p2(3) - p1(3))*(p3(1) - p1(1)) - (p2(1) - p1(1))*(p3(3) - p1(3));
  c = (p2(1) - p1(1))*(p3(2) - p1(2)) - (p2(2) - p1(2))*(p3(1) - p1(1));
  d = - p2(1)*a - p2(2)*b - p2(3)*c;
  return
end
\end{varbatim}
}
\subsection{Plane\_std\_point\_dist\_3d}
{\small
\begin{verbatim}
function [dist, p_0] = plane_std_point_dist_3d ( a, b, c, d, p )
%Plane_std_point_dist_3d determines the distance between a set of points
%and a plane when the plane is in the following form:
%
%   STANDARD FORM IS
%   A * X + B * Y + C * Z + D = 0;
%
%Also determines the points projected on the plane.
%   pp_x = x_0+t*a;
%   pp_y = y_0+t*b;
%   pp_z = z_0+t*c;
%
%   PLANE_EXP2IMP_3D(P1,P2,P3)
%       DIST: DISTANCE BETWEEN POINT AND PLANE (NX1)
%       P_0: PROJECTED POINTS (NX3)
%       A: COEFFICIENT A
%       B: COEFFICIENT B
%       C: COEFFICIENT C
%       D: COEFFICIENT D
%       P: SET OF POINTS (NX3)
%--------------------------------------------------------------------------
%   Variables:
%       NORM: "NORMAL" OF THE COEFFICIENTS
%       T_0: PARAMETER T
  norm = sqrt ( a * a + b * b + c * c );
  t_0=-(p(:,1)*a + p(:,2)*b + p(:,3)*c + ones(size(p,1),1)*d)./( a * a + b * b + c * c );
```

```
  p_0=[(p(:,1)+t_0*a),(p(:,2)+t_0*b),(p(:,3)+t_0*c)];
  dist = abs ( p(:,1)*a + p(:,2)*b + p(:,3)*c + ones(size(p,1),1)*d ) ./ norm;
  return
end
```

B.3 2D Processing Code

```
function [ glob_K,glob_V,glob_E,enodes ] = FEM_2D_QW(p,t,vert,V_nodes)
%FEM_2D_QW This is finite element code that populates the global matrices
%needed to solve Schrodinger's equation for a quantum wire (2D).
%
%
%   FEM_2D_QW(P,T,VERT,V_NODES)
%       GLOB_K: KINETIC ENERGY MATRIX (SPARSE MATRIX)
%       GLOB_V: POTENTIAL ENERGY MATRIX (SPARSE MATRIX)
%       GLOB_E: OVERLAP MATRIX (SPARSE MATRIX)
%       ENODES: NODES ON THE BOUNDING BOX (NEEDED FOR POST PROCESSING)
%       P: COORDINATES OF NODES (NNODES X 2)
%       T: TRIANGULAR ELEMENTS (NEL X NNEL)
%       VERT: VERTICES OF QUANTUM WIRE (M X P,M VERTICES IN P DIMENSIONS)
%       V_NODES: POTENTIAL AT EACH NODE OTHER THAN OFFSET IN eV (NNODES X 1)
%-------------------------------------------------------------------------
%
%   Physical constants:
%       m_e: mass of electron in grams
%       h_bar: plancks constant in kilograms-Angstrom per second
%       q: measure of electron-volt in gram-centimeter squared per second
%       squared
%   Variables:
%       ndof: number of degrees of freedom (dof) per node (1 OR 3)
%       m_eff: effective mass of semiconductor in grams(1 X 2)
%       V_offset: conduction or valence band offset in eV (1 X 2)
%       nnel: number of nodes per element
%       nel: number of elements
%       nnodes: number of nodes
%       interface_nodes: nodes on interface separating the two materials
%       elem_size: size of element matrices
%       r,c,v: row & column indices of respective element matrix that point
%           to value v.
%       nd: local node indexing (1 X NNEL)
%       xcoord/ycoord: local coordinate indexing (1 X NNEL)
%       V: potential energy per node (1 X NNEL)
%       m: effective mass of element
%       J: Jacobian Matrix (2 X 2)
%       Elem_K: elemental kinetic matrix (EDOF X EDOF)
%       Elem_V: elemental potential matrix (EDOF X EDOF)
%       Elem_E: elemental overlap matrix (EDOF X EDOF)
%       index: system degree of freedom assigned to element node(1 X EDOF)
%   Functions:
%       boundedges: Find boundary edges from mesh
%       inhull: Tests if a set of points is inside a convex hull
%       jacob_2D: The Jacobian for 2D mapping
%       Elem_matrix_K: Computes the elemental kinetic matrix
%       Elem_matrix_V: Computes the elemental potential matrix
%       Elem_matrix_E: Computes the elemental overlap matrix
%       apply_interface_bc_2D: Applies the boundary conditions on the
%           interface between the two materials if dof > 1.
%       index_2D: Assigns the system dof to the element node
```

```
%       assemble: Builds up row & column indices
%       sparse: assembles global matrices by creating a sparse matrix
%
m_e = 9.109*10^-28;
%ENTER USER INPUT HERE-----------------------------------------------------
ndof = 3;
%(1) represents the well and (2) represents the barrier
m_eff(1)=m_e*0.0665;
m_eff(2)=m_e*0.0858;
V_offset(1)=0;
V_offset(2)=0.276;
%END USER INPUT-------------------------------------------------------------
%
h_bar = 1.054*10^-19;
q = 1.602*10^-12;
nnel = 3;
nel=length(t(:,1));
nnode=length(p(:,1));
interface_nodes = find(ismember(p,vert,'rows'));
%
%For bound state energies, which this program is solving for, boundary
%conditions require the wavefunction to fall off at regions approaching
%infinity.  In order to satisfy this, we bound the barrier region at
%"sufficient" distances.
%
enodes = unique(boundedges(p,t));
%
elem_size = nnel*ndof*nnel*ndof;
r_K = zeros(nel,elem_size);r_V = zeros(nel,elem_size);r_E = zeros(nel,elem_size);
c_K = zeros(nel,elem_size);c_V = zeros(nel,elem_size);c_E = zeros(nel,elem_size);
v_K = zeros(nel,elem_size);v_V = zeros(nel,elem_size);v_E = zeros(nel,elem_size);
%
for iel=1:nel
    nd = t(iel,:);
    xcoord = p(nd,1);
    ycoord = p(nd,2);
    IN = inhull([xcoord ycoord],vert,[],0.01);
    if  (sum(IN) == nnel)
        V(1:nnel) = V_offset(1) + V_nodes(nd);
        m = m_eff(1);
    else
        V(1:nnel) = V_offset(2) + V_nodes(nd);
        m = m_eff(2);
    end
    J = jacob_2D(xcoord,ycoord);
    %
    %Construct the elemental matrices by using functions Elem_matrix_K,
    %Elem_matrix_V, and Elem_matrix_E.
    elem_K = ((h_bar)^2/(2*m*q))*Elem_matrix_K(J,ndof);
    elem_V = Elem_matrix_V(J,V,ndof);
    elem_E = Elem_matrix_E(J,ndof);
    %
    %Add reciprical mass condition at interface if there is more than 1
    %dof per node.
    if (ndof > 1 && m == m_eff(1))
        elem_K = apply_interface_bc_2D(elem_K,interface_nodes,nd,m_eff(2)/m);
        elem_V = apply_interface_bc_2D(elem_V,interface_nodes,nd,m_eff(2)/m);
        elem_E = apply_interface_bc_2D(elem_E,interface_nodes,nd,m_eff(2)/m);
    end
```

```
    %
    %Builds up row & column indices using assemble function
    %
    index = index_2D(nd,ndof);
    [r_K(iel,:) c_K(iel,:) v_K(iel,:)] = assemble(elem_K,index);
    [r_V(iel,:) c_V(iel,:) v_V(iel,:)] = assemble(elem_V,index);
    [r_E(iel,:) c_E(iel,:) v_E(iel,:)] = assemble(elem_E,index);
end
%
%Construct the global matrices using sparse and triples of
%rows/columns/values
%
glob_K = sparse(r_K,c_K,v_K,ndof*nnode,ndof*nnode);
glob_V = sparse(r_V,c_V,v_V,ndof*nnode,ndof*nnode);
glob_E = sparse(r_E,c_E,v_E,ndof*nnode,ndof*nnode);
end
```

B.3.1 Jacob_2D

```
function [ J ] = jacob_2D( x,y )
%Jacob_2D The Jacobian for the 2D mapping
%   The Jacobian is constant due to the linear transformation of local
%   coordinates to global coordinates.  Linear shape functions are used as
%   the coordinate transformation.
%
%   [J]=JACOB_2D(X,Y)
%      J: JACOBIAN MATRIX (2x2)
%      X: GLOBAL COORDINATES IN THE X DIRECTION(1x3)
%      Y: GLOBAL COORDINATES IN THE Y DIRECTION(1x3)
%--------------------------------------------------------------------------
%
J(1,1) = x(2)-x(1);
J(1,2) = y(2)-y(1);
J(2,1) = x(3)-x(1);
J(2,2) = y(3)-y(1);
end
```

B.3.2 Elem_matrix_E

```
function [ elem_E ] = Elem_matrix_E( J,ndof )
%Elem_matrix_E Computes the elemental overlap matrix
%   The elemental overlap energy matrix is formed using either linear
%   basis functions or cubic basis functions.
%
%   [ELEM_E]=ELEM_MATRIX_E(J,NDOF)
%      ELEM_E: ELEMENTAL OVERLAP ENERGY MATRIX (3 X 3 OR 9 X 9)
%      J: JACOBIAN MATRIX (2 X 2)
%      NDOF: DEGREE OF FREEDOM PER NODE
%--------------------------------------------------------------------------
%
%   Variables:
%       Translate: Matrix that converts ELEM_E from local coordinates to
%       global coordinates.  This is only used when ndof does not equal 1.
%
%Linear basis functions are defined as;
%   N_1(x,y)=1-x-y;     0<=x,y<=1
%   N_2(x,y)=x;         0<=x,y<=1
%   N_3(x,y)=y;         0<=x,y<=1
%
```

```
%Cubic basis functions are defined as;
%   N_1(x,y)=1-3*x^2-3*y^2+2*x^3+2*y^3;                    0<=x,y<=1
%   N_2(x,y)=x-2*x^2+x^3+1/3*(-x*y+x^2*y-2*x*y^2);         0<=x,y<=1
%   N_3(x,y)=y-2*y^2+y^3+1/3*(-x*y+x*y^2-2*x^2*y);         0<=x,y<=1
%   N_4(x,y)=3*x^2-2*x^3;                                  0<=x,y<=1
%   N_5(x,y)=-x^2+x^3;                                     0<=x,y<=1
%   N_6(x,y)=1/3*(x*y+2*x^2*y-x*y^2);                      0<=x,y<=1
%   N_7(x,y)=3*y^2-2*y^3;                                  0<=x,y<=1
%   N_8(x,y)=1/3*(x*y+2*x*y^2-x^2*y);                      0<=x,y<=1
%   N_9(x,y)=-y^2+y^3;                                     0<=x,y<=1
%
%-------------------------------------------------------------------------
%
if (ndof == 1)
    elem_E = 1/24*[2 1 1 ; 1 2 1 ; 1 1 2];
    elem_E = det(J)*elem_E;
else
    elem_E = [17/140, 9/560, 9/560, 11/280, -17/1680, 1/210, 11/280, 1/210,...
              -17/1680 ; 9/560, 31/11340, 97/45360, 11/1260, -11/5040,...
              13/12960, 29/5040, 41/45360, -1/672 ; 9/560, 97/45360,...
              31/11340, 29/5040, -1/672, 41/45360, 11/1260, 13/12960,...
              -11/5040 ; 11/280, 11/1260, 29/5040, 3/35, -1/60, 23/2520,...
              1/40, 1/180, -11/1680 ; -17/1680, -11/5040, -1/672, -1/60,...
              1/280, -1/504, -11/1680, -1/720, 17/10080 ; 1/210, 13/12960,...
              41/45360, 23/2520, -1/504, 4/2835, 1/180, 101/90720, -1/720 ;...
              11/280, 29/5040, 11/1260, 1/40, -11/1680, 1/180, 3/35, 23/2520,...
              -1/60 ; 1/210, 41/45360, 13/12960, 1/180, -1/720, 101/90720,...
              23/2520, 4/2835, -1/504 ; -17/1680, -1/672, -11/5040, -11/1680,...
              17/10080, -1/720, -1/60, -1/504, 1/280];
    translate = zeros(9,9);
    translate(1,1) = 1;
    translate(4,4) = 1;
    translate(7,7) = 1;
    translate(2:3,2:3) = J;
    translate(5:6,5:6) = J;
    translate(8:9,8:9) = J;
    elem_E = det(J)*(translate'*elem_E*translate);
end
```

B.3.3 Elem_matrix_V

```
function [ elem_V ] = Elem_matrix_V( J,V,ndof )
%Elem_matrix_V Computes the elemental potential matrix
%   The elemental potential energy matrix is formed using either linear
%   basis functions or cubic basis functions.
%
%   [ELEM_V]=ELEM_MATRIX_V(J,V,NDOF)
%      ELEM_V: ELEMENTAL POTENTIAL ENERGY MATRIX (3 X 3 OR 9 X 9)
%      J: JACOBIAN MATRIX (2 X 2)
%      V: VECTOR CONTAINING POTENTIAL AT NODE (1 X 3)
%      NDOF: DEGREE OF FREEDOM PER NODE
%-------------------------------------------------------------------------
%
%   Variables:
%       Translate: Matrix that converts ELEM_V from local coordinates to
%       global coordinates.  This is only used when ndof does not equal 1.
%
%Linear shape functions are defined as;
%   N_1(x,y)=1-x-y;     0<=x,y<=1
```

```
%   N_2(x,y)=x;         0<=x,y<=1
%   N_3(x,y)=y;         0<=x,y<=1
%
%Cubic shape functions are defined as;
%   N_1(x,y)=1-3*x^2-3*y^2+2*x^3+2*y^3;                    0<=x,y<=1
%   N_2(x,y)=x-2*x^2+x^3+1/3*(-x*y+x^2*y-2*x*y^2);         0<=x,y<=1
%   N_3(x,y)=y-2*y^2+y^3+1/3*(-x*y+x*y^2-2*x^2*y);         0<=x,y<=1
%   N_4(x,y)=3*x^2-2*x^3;                                  0<=x,y<=1
%   N_5(x,y)=-x^2+x^3;                                     0<=x,y<=1
%   N_6(x,y)=1/3*(x*y+2*x^2*y-x*y^2);                      0<=x,y<=1
%   N_7(x,y)=3*y^2-2*y^3;                                  0<=x,y<=1
%   N_8(x,y)=1/3*(x*y+2*x*y^2-x^2*y);                      0<=x,y<=1
%   N_9(x,y)=-y^2+y^3;                                     0<=x,y<=1
%
%-------------------------------------------------------------------------
%
if (ndof == 1)
    elem_V = 1/120*[6*V(1)+2*V(2)+2*V(3) 2*V(1)+2*V(2)+1*V(3) 2*V(1)+1*V(2)+2*V(3) ; ...
            2*V(1)+2*V(2)+1*V(3) 2*V(1)+6*V(2)+2*V(3) 1*V(1)+2*V(2)+2*V(3); ...
            2*V(1)+1*V(2)+2*V(3) 1*V(1)+2*V(2)+2*V(3) 2*V(1)+2*V(2)+6*V(3)];
    elem_V = det(J)*elem_V;
else
    elem_V = [(1/14)*V(1)+(1/40)*V(3)+(1/40)*V(2),...
              (293/90720)*V(3)+(59/12960)*V(2)+(47/5670)*V(1),...
              (47/5670)*V(1)+(293/90720)*V(2)+(59/12960)*V(3),...
              (1/70)*V(1)+(13/1680)*V(3)+(29/1680)*V(2),...
              -(13/3360)*V(1)-(1/480)*V(3)-(1/240)*V(2),...
              (139/90720)*V(1)+(127/90720)*V(3)+(83/45360)*V(2),...
              (1/70)*V(1)+(13/1680)*V(2)+(29/1680)*V(3),...
              (139/90720)*V(1)+(83/45360)*V(3)+(127/90720)*V(2),...
              -(13/3360)*V(1)-(1/480)*V(2)-(1/240)*V(3) ;...
              (293/90720)*V(3)+(59/12960)*V(2)+(47/5670)*V(1),...
              (73/136080)*V(3)+(31/34020)*V(2)+(5/3888)*V(1),...
              (13/13608)*V(1)+(23/38880)*V(2)+(23/38880)*V(3),...
              (19/11340)*V(3)+(187/45360)*V(2)+(19/6480)*V(1),...
              -(1/1296)*V(1)-(1/2268)*V(3)-(11/11340)*V(2),...
              (79/272160)*V(1)+(23/54432)*V(2)+(79/272160)*V(3),...
              (79/45360)*V(1)+(137/90720)*V(2)+(227/90720)*V(3),...
              (1/3888)*V(1)+(41/136080)*V(2)+(47/136080)*V(3),...
              -(43/90720)*V(1)-(37/90720)*V(2)-(11/18144)*V(3) ;...
              (47/5670)*V(1)+(293/90720)*V(2)+(59/12960)*V(3),...
              (13/13608)*V(1)+(23/38880)*V(2)+(23/38880)*V(3),...
              (5/3888)*V(1)+(73/136080)*V(2)+(31/34020)*V(3),...
              (79/45360)*V(1)+(227/90720)*V(2)+(137/90720)*V(3),...
              -(43/90720)*V(1)-(11/18144)*V(2)-(37/90720)*V(3),...
              (47/136080)*V(2)+(41/136080)*V(3)+(1/3888)*V(1),...
              (19/6480)*V(1)+(187/45360)*V(3)+(19/11340)*V(2),...
              (79/272160)*V(1)+(79/272160)*V(2)+(23/54432)*V(3),...
              -(1/1296)*V(1)-(11/11340)*V(3)-(1/2268)*V(2) ;...
              (1/70)*V(1)+(13/1680)*V(3)+(29/1680)*V(2),...
              (19/11340)*V(3)+(187/45360)*V(2)+(19/6480)*V(1),...
              (79/45360)*V(1)+(227/90720)*V(2)+(137/90720)*V(3),...
              (19/1260)*V(1)+(1/18)*V(2)+(19/1260)*V(3),...
              -(17/5040)*V(1)-(5/504)*V(2)-(17/5040)*V(3),...
              (121/90720)*V(1)+(121/22680)*V(2)+(223/90720)*V(3),...
              (1/252)*V(1)+(53/5040)*V(2)+(53/5040)*V(3),...
              (19/22680)*V(1)+(17/6480)*V(2)+(19/9072)*V(3),...
              -(11/10080)*V(1)-(29/10080)*V(2)-(13/5040)*V(3) ;...
              -(13/3360)*V(1)-(1/480)*V(3)-(1/240)*V(2),...
```

```
            -(1/1296)*V(1)-(1/2268)*V(3)-(11/11340)*V(2),...
            -(43/90720)*V(1)-(11/18144)*V(2)-(37/90720)*V(3),...
            -(17/5040)*V(1)-(5/504)*V(2)-(17/5040)*V(3),...
            (1/1260)*V(1)+(1/504)*V(2)+(1/1260)*V(3),...
            -(29/90720)*V(1)-(7/6480)*V(2)-(53/90720)*V(3),...
            -(11/10080)*V(1)-(13/5040)*V(2)-(29/10080)*V(3),...
            -(1/4536)*V(1)-(1/1620)*V(2)-(5/9072)*V(3),...
            (1/3360)*V(1)+(1/1440)*V(2)+(1/1440)*V(3) ;...
            (139/90720)*V(1)+(127/90720)*V(3)+(83/45360)*V(2),...
            (79/272160)*V(1)+(23/54432)*V(2)+(79/272160)*V(3),...
            (47/136080)*V(2)+(41/136080)*V(3)+(1/3888)*V(1),...
            (121/90720)*V(1)+(121/22680)*V(2)+(223/90720)*V(3),...
            -(29/90720)*V(1)-(7/6480)*V(2)-(53/90720)*V(3),...
            (5/27216)*V(1)+(103/136080)*V(2)+(4/8505)*V(3),...
            (19/22680)*V(1)+(19/9072)*V(2)+(17/6480)*V(3),...
            (41/272160)*V(1)+(131/272160)*V(2)+(131/272160)*V(3),...
            -(1/4536)*V(1)-(5/9072)*V(2)-(1/1620)*V(3) ;...
            (1/70)*V(1)+(13/1680)*V(2)+(29/1680)*V(3),...
            (79/45360)*V(1)+(137/90720)*V(2)+(227/90720)*V(3),...
            (19/6480)*V(1)+(187/45360)*V(3)+(19/11340)*V(2),...
            (1/252)*V(1)+(53/5040)*V(2)+(53/5040)*V(3),...
            -(11/10080)*V(1)-(13/5040)*V(2)-(29/10080)*V(3),...
            (19/22680)*V(1)+(19/9072)*V(2)+(17/6480)*V(3),...
            (19/1260)*V(1)+(19/1260)*V(2)+(1/18)*V(3),...
            (121/90720)*V(1)+(223/90720)*V(2)+(121/22680)*V(3),...
            -(17/5040)*V(1)-(17/5040)*V(2)-(5/504)*V(3) ;...
            (139/90720)*V(1)+(83/45360)*V(3)+(127/90720)*V(2),...
            (1/3888)*V(1)+(41/136080)*V(2)+(47/136080)*V(3),...
            (79/272160)*V(1)+(79/272160)*V(2)+(23/54432)*V(3),...
            (19/22680)*V(1)+(17/6480)*V(2)+(19/9072)*V(3),...
            -(1/4536)*V(1)-(1/1620)*V(2)-(5/9072)*V(3),...
            (41/272160)*V(1)+(131/272160)*V(2)+(131/272160)*V(3),...
            (121/90720)*V(1)+(223/90720)*V(2)+(121/22680)*V(3),...
            (5/27216)*V(1)+(4/8505)*V(2)+(103/136080)*V(3),...
            -(29/90720)*V(1)-(53/90720)*V(2)-(7/6480)*V(3) ;...
            -(13/3360)*V(1)-(1/480)*V(2)-(1/240)*V(3),...
            -(43/90720)*V(1)-(37/90720)*V(2)-(11/18144)*V(3),...
            -(1/1296)*V(1)-(11/11340)*V(3)-(1/2268)*V(2),...
            -(11/10080)*V(1)-(29/10080)*V(2)-(13/5040)*V(3),...
            (1/3360)*V(1)+(1/1440)*V(2)+(1/1440)*V(3),...
            -(1/4536)*V(1)-(5/9072)*V(2)-(1/1620)*V(3),...
            -(17/5040)*V(1)-(17/5040)*V(2)-(5/504)*V(3),...
            -(29/90720)*V(1)-(53/90720)*V(2)-(7/6480)*V(3),...
            (1/1260)*V(1)+(1/1260)*V(2)+(1/504)*V(3)];
    translate = zeros(9,9);
    translate(1,1) = 1;
    translate(4,4) = 1;
    translate(7,7) = 1;
    translate(2:3,2:3) = J;
    translate(5:6,5:6) = J;
    translate(8:9,8:9) = J;
    elem_V = det(J)*(translate'*elem_V*translate);
end
```

B.3.4 Elem_matrix_K

```
function [ elem_K ] = Elem_matrix_K( J,ndof )
%Elem_matrix_K Computes the elemental kinetic matrix
%   The elemental kinetic energy matrix is formed using either linear basis
```

```
%   functions or cubic basis functions.
%
%   [ELEM_K]=ELEM_MATRIX_K(J,NDOF)
%      ELEM_K: ELEMENTAL KINETIC ENERGY MATRIX (3 X 3 OR 9 X 9)
%      J: JACOBIAN MATRIX(2 X 2)
%      NDOF: DEGREE OF FREEDOM PER NODE
%-------------------------------------------------------------------------
%The elemental kinetic energy matrix is constructed using the quadratic
%form as represented by the equation Trace(M_C*(JJ^T)^-1).  This form is
%convenient because the integrals are carried out once and represented in
%matricies M_C, Mxx, Mxy, Myx, and Myy below.
%   Variables:
%       M_C: MATRIX REPRESENTING THE PRODUCT OF DERIVATIVES (2 X 2)
%       MXX: VECTOR REPRESENTING PRODUCT OF SHAPE FUNCTION DERIVATIVES WITH
%       RESPECT TO X (1 X 9 OR 1 X 81)
%       MXY: VECTOR REPRESENTING PRODUCT OF SHAPE FUNCTION DERIVATIVES WITH
%       RESPECT TO X AND Y (1 X 9 OR 1 X 81)
%       MYX: VECTOR REPRESENTING PRODUCT OF SHAPE FUNCTION DERIVATIVES WITH
%       RESPECT TO Y AND X (1 X 9 OR 1 X 81)
%       MYY: VECTOR REPRESENTING PRODUCT OF SHAPE FUNCTION DERIVATIVES WITH
%       RESPECT TO Y (1 X 9 OR 1 X 81)
%       TRANSLATE: Matrix that converts ELEM_K from local coordinates to
%       global coordinates.  This is only used when ndof does not equal 1.
%
%Linear basis functions are defined as;
%   N_1(x,y)=1-x-y;     0<=x,y<=1
%   N_2(x,y)=x;         0<=x,y<=1
%   N_3(x,y)=y;         0<=x,y<=1
%
%Cubic basis functions are defined as;
%   N_1(x,y)=1-3*x^2-3*y^2+2*x^3+2*y^3;                     0<=x,y<=1
%   N_2(x,y)=x-2*x^2+x^3+1/3*(-x*y+x^2*y-2*x*y^2);          0<=x,y<=1
%   N_3(x,y)=y-2*y^2+y^3+1/3*(-x*y+x*y^2-2*x^2*y);          0<=x,y<=1
%   N_4(x,y)=3*x^2-2*x^3;                                   0<=x,y<=1
%   N_5(x,y)=-x^2+x^3;                                      0<=x,y<=1
%   N_6(x,y)=1/3*(x*y+2*x^2*y-x*y^2);                       0<=x,y<=1
%   N_7(x,y)=3*y^2-2*y^3;                                   0<=x,y<=1
%   N_8(x,y)=1/3*(x*y+2*x*y^2-x^2*y);                       0<=x,y<=1
%   N_9(x,y)=-y^2+y^3;                                      0<=x,y<=1
%
%-------------------------------------------------------------------------
%
K = (J*J')^-1;
if (ndof == 1)
    elem_K = zeros(3,3);
    Mxx = [1/2 -1/2 0  -1/2 1/2 0  0 0 0];
    Mxy = [1/2 0 -1/2  -1/2 0 1/2  0 0 0];
    Myx = [1/2 -1/2 0  0 0 0  -1/2 1/2 0];
    Myy = [1/2 0 -1/2  0 0 0  -1/2 0 1/2];
    M = [Mxx ; Mxy ; Myx ; Myy];
    M_C1 = reshape(M(:,1),2,2)'*K ; M_C2 = reshape(M(:,2),2,2)'*K;
    M_C3 = reshape(M(:,3),2,2)'*K ; M_C5 = reshape(M(:,5),2,2)'*K;
    M_C6 = reshape(M(:,6),2,2)'*K ; M_C9 = reshape(M(:,9),2,2)'*K;
    elem_K(1,1) = trace(M_C1); elem_K(1,2) = trace(M_C2);
    elem_K(1,3) = trace(M_C3); elem_K(2,2) = trace(M_C5);
    elem_K(2,3) = trace(M_C6); elem_K(3,3) = trace(M_C9);
    %
    elem_K = elem_K + elem_K' - diag(diag(elem_K));
```

```
    elem_K = det(J)*elem_K;
else
    elem_K = zeros(9,9);
    Mxx = [3/5, 11/180, 17/180, -3/5, 1/10, -17/180, 0, -11/180, 0 ...
        11/180, 91/1620, 4/405, -11/180, 0, -4/405, 0, 2/405, 0 ...
        17/180, 4/405, 31/1620, -17/180, 1/60, -31/1620, 0, -5/324, 0 ...
        -3/5, -11/180, -17/180, 3/5, -1/10, 17/180, 0, 11/180, 0 ...
        1/10, 0, 1/60, -1/10, 1/30, -1/60, 0, -1/60, 0 ...
        -17/180, -4/405, -31/1620, 17/180, -1/60, 31/1620, 0, 5/324, 0 ...
        0, 0, 0, 0, 0, 0, 0, 0, 0 ...
        -11/180, 2/405, -5/324, 11/180, -1/60, 5/324, 0, 11/324, 0 ...
        0, 0, 0, 0, 0, 0, 0, 0, 0];
    Mxy = [1/2, 1/10, 0, 0, 0, -1/10, -1/2, -1/10, 1/10 ...
        0, 43/3240, 61/3240, 0, 0, -61/3240, 0, -43/3240, 0 ...
        1/10, 61/3240, 43/3240, 0, 0, -43/3240, -1/10, -61/3240, 1/60 ...
        -1/2, -1/10, 0, 0, 0, 1/10, 1/2, 1/10, -1/10 ...
        1/10, 1/60, 0, 0, 0, 0, -1/10, -1/60, 1/60 ...
        -1/10, -61/3240, -43/3240, 0, 0, 43/3240, 1/10, 61/3240, -1/60 ...
        0, 0, 0, 0, 0, 0, 0, 0, 0 ...
        -1/10, -43/3240, -61/3240, 0, 0, 7/3240, 1/10, 43/3240, 0 ...
        0, 0, 0, 0, 0, 0, 0, 0, 0];
    Myx = [1/2, 0, 1/10, -1/2, 1/10, -1/10, 0, -1/10, 0 ...
        1/10, 43/3240, 61/3240, -1/10, 1/60, -61/3240, 0, -43/3240, 0 ...
        0, 61/3240, 43/3240, 0, 0, -43/3240, 0, -61/3240, 0 ...
        0, 0, 0, 0, 0, 0, 0, 0, 0 ...
        0, 0, 0, 0, 0, 0, 0, 0, 0 ...
        -1/10, -61/3240, -43/3240, 1/10, 0, 43/3240, 0, 7/3240, 0 ...
        -1/2, 0, -1/10, 1/2, -1/10, 1/10, 0, 1/10, 0 ...
        -1/10, -43/3240, -61/3240, 1/10, -1/60, 61/3240, 0, 43/3240, 0 ...
        1/10, 0, 1/60, -1/10, 1/60, -1/60, 0, 0, 0];
    Myy = [3/5, 17/180, 11/180, 0, 0, -11/180, -3/5, -17/180, 1/10 ...
        17/180, 31/1620, 4/405, 0, 0, -5/324, -17/180, -31/1620, 1/60 ...
        11/180, 4/405, 91/1620, 0, 0, 2/405, -11/180, -4/405, 0 ...
        0, 0, 0, 0, 0, 0, 0, 0, 0 ...
        0, 0, 0, 0, 0, 0, 0, 0, 0 ...
        -11/180, -5/324, 2/405, 0, 0, 11/324, 11/180, 5/324, -1/60 ...
        -3/5, -17/180, -11/180, 0, 0, 11/180, 3/5, 17/180, -1/10 ...
        -17/180, -31/1620, -4/405, 0, 0, 5/324, 17/180, 31/1620, -1/60 ...
        1/10, 1/60, 0, 0, 0, -1/60, -1/10, -1/60, 1/30];
    M = [Mxx ; Mxy ; Myx ; Myy];
    M_C1 = reshape(M(:,1),2,2)'*K ; M_C2 = reshape(M(:,2),2,2)'*K;
    M_C3 = reshape(M(:,3),2,2)'*K ; M_C4 = reshape(M(:,4),2,2)'*K;
    M_C5 = reshape(M(:,5),2,2)'*K ; M_C6 = reshape(M(:,6),2,2)'*K;
    M_C7 = reshape(M(:,7),2,2)'*K ; M_C8 = reshape(M(:,8),2,2)'*K;
    M_C9 = reshape(M(:,9),2,2)'*K ; M_C11 = reshape(M(:,11),2,2)'*K;
    M_C12 = reshape(M(:,12),2,2)'*K ; M_C13 = reshape(M(:,13),2,2)'*K;
    M_C14 = reshape(M(:,14),2,2)'*K ; M_C15 = reshape(M(:,15),2,2)'*K;
    M_C16 = reshape(M(:,16),2,2)'*K ; M_C17 = reshape(M(:,17),2,2)'*K;
    M_C18 = reshape(M(:,18),2,2)'*K ; M_C21 = reshape(M(:,21),2,2)'*K;
    M_C22 = reshape(M(:,22),2,2)'*K ; M_C23 = reshape(M(:,23),2,2)'*K;
    M_C24 = reshape(M(:,24),2,2)'*K ; M_C25 = reshape(M(:,25),2,2)'*K;
    M_C26 = reshape(M(:,26),2,2)'*K ; M_C27 = reshape(M(:,27),2,2)'*K;
    M_C31 = reshape(M(:,31),2,2)'*K ; M_C32 = reshape(M(:,32),2,2)'*K;
    M_C33 = reshape(M(:,33),2,2)'*K ; M_C34 = reshape(M(:,34),2,2)'*K;
    M_C35 = reshape(M(:,35),2,2)'*K ; M_C36 = reshape(M(:,36),2,2)'*K;
    M_C41 = reshape(M(:,41),2,2)'*K ; M_C42 = reshape(M(:,42),2,2)'*K;
    M_C43 = reshape(M(:,43),2,2)'*K ; M_C44 = reshape(M(:,44),2,2)'*K;
    M_C45 = reshape(M(:,45),2,2)'*K ; M_C51 = reshape(M(:,51),2,2)'*K;
    M_C52 = reshape(M(:,52),2,2)'*K ; M_C53 = reshape(M(:,53),2,2)'*K;
```

```
    M_C54 = reshape(M(:,54),2,2)'*K ; M_C61 = reshape(M(:,61),2,2)'*K;
    M_C62 = reshape(M(:,62),2,2)'*K ; M_C63 = reshape(M(:,63),2,2)'*K;
    M_C71 = reshape(M(:,71),2,2)'*K ; M_C72 = reshape(M(:,72),2,2)'*K;
    M_C81 = reshape(M(:,81),2,2)'*K;
    %
    elem_K(1,1) = trace(M_C1) ; elem_K(1,2) = trace(M_C2);
    elem_K(1,3) = trace(M_C3) ; elem_K(1,4) = trace(M_C4);
    elem_K(1,5) = trace(M_C5) ; elem_K(1,6) = trace(M_C6);
    elem_K(1,7) = trace(M_C7) ; elem_K(1,8) = trace(M_C8);
    elem_K(1,9) = trace(M_C9) ; elem_K(2,2) = trace(M_C11);
    elem_K(2,3) = trace(M_C12) ; elem_K(2,4) = trace(M_C13);
    elem_K(2,5) = trace(M_C14) ; elem_K(2,6) = trace(M_C15);
    elem_K(2,7) = trace(M_C16) ; elem_K(2,8) = trace(M_C17);
    elem_K(2,9) = trace(M_C18) ; elem_K(3,3) = trace(M_C21);
    elem_K(3,4) = trace(M_C22) ; elem_K(3,5) = trace(M_C23);
    elem_K(3,6) = trace(M_C24) ; elem_K(3,7) = trace(M_C25);
    elem_K(3,8) = trace(M_C26) ; elem_K(3,9) = trace(M_C27);
    elem_K(4,4) = trace(M_C31) ; elem_K(4,5) = trace(M_C32);
    elem_K(4,6) = trace(M_C33) ; elem_K(4,7) = trace(M_C34);
    elem_K(4,8) = trace(M_C35) ; elem_K(4,9) = trace(M_C36);
    elem_K(5,5) = trace(M_C41) ; elem_K(5,6) = trace(M_C42);
    elem_K(5,7) = trace(M_C43) ; elem_K(5,8) = trace(M_C44);
    elem_K(5,9) = trace(M_C45) ; elem_K(6,6) = trace(M_C51);
    elem_K(6,7) = trace(M_C52) ; elem_K(6,8) = trace(M_C53);
    elem_K(6,9) = trace(M_C54) ; elem_K(7,7) = trace(M_C61);
    elem_K(7,8) = trace(M_C62) ; elem_K(7,9) = trace(M_C63);
    elem_K(8,8) = trace(M_C71) ; elem_K(8,9) = trace(M_C72);
    elem_K(9,9) = trace(M_C81) ;
    %
    translate = zeros(9,9);
    translate(1,1) = 1;
    translate(4,4) = 1;
    translate(7,7) = 1;
    translate(2:3,2:3) = J;
    translate(5:6,5:6) = J;
    translate(8:9,8:9) = J;
    elem_K = elem_K + elem_K' - diag(diag(elem_K));
    elem_K = det(J)*(translate'*elem_K*translate);
end
```

B.3.5 Apply_interface_bc_2D

```
function [ elem_A ] = apply_interface_bc_2D( elem_A,int_nodes,nd,m )
%Apply_interface_bc Applies the boundary conditions on the interface
%between the two materials.
%
%   [GLOB_A]=APPLY_BC(GLOB_A,INT_NODES,ND,M)
%      Elem_A: ELEMENTAL MATRIX
%      INT_NODES: VECTOR CONTAINING NODES ON THE INTERFACE BETWEEN THE TWO
%      MATERIALS
%      ND: LOCAL OR ELEMENTAL NODE INDEXING (1 X 4)
%      M: EFFECTIVE MASS
%-------------------------------------------------------------------------
ind = find(ismember(nd,int_nodes));
if (~isempty(ind))
    k=0;
    for i=1:length(ind)
    start=(ind(i)-1)*3;
        for j=2:3;
```

```
            k=k+1;
            e_index(k)=start+j;
        end
    end
    elem_A(:,e_index) = 1/m*elem_A(:,e_index);
    elem_A(e_index,:) = 1/m*elem_A(e_index,:);
end
end
```

B.3.6 Index_2D

```
function [ index ] = index_2D( nd,ndof )
%Index_2D Assigns the system degree of freedom to the element node
%   Index is system dof vector which can be used to place the elements
%   associated with element matrices in the global matrices.
%
%   [INDEX]=INDEX_3D(ND,NNEL,NDOF)
%      INDEX: SYSTEM DOF VECTOR
%      ND: VECTOR CONTAINING GLOBAL NODE NUMBERS(1 X 4)
%      NDOF: DEGREE OF FREEDOM PER NODE
%--------------------------------------------------------------------------
%
start=(nd-1)*ndof+1;
if (ndof == 1)
    index = start;
else
    index = [start ; start+1 ; start+2];
    index = index(:)';
end
```

B.3.7 Assemble

```
function [ row column value ] = assemble( A,index )
%Assemble Store global matrix information in row, column,
%and value format.
%
%   [ROW COLUMN VALUE]=ASSEMBLE(A,INDEX)
%      ROW: ROW INDEX CORRESPONDING TO EACH VALUE OF MATRIX ELEMENT
%      COLUMN: COLUMN INDEX CORRESPONDING TO EACH VALUE OF MATRIX ELEMENT
%      VALUE: MATRIX ELEMENT
%      A: ELEMENT MATRIX
%      INDEX: SYSTEM DOF VECTOR
%--------------------------------------------------------------------------
%
[row column] = meshgrid(index,index);
row = row(:)';
column = column(:)';
value = A(:)';
end
```

B.4 3D Processing Code

```
function [ glob_K,glob_V,glob_E,enodes ] = FEM_3D_QD(p,t,vert,V_nodes)
%FEM_2D_QW This is finite element code that populates the global matrices
%needed to solve Schrodinger's equation for a quantum dot (3D).
%
%
%   FEM_3D_QD(P,T,VERT,V_NODES)
```

```
%       GLOB_K: KINETIC ENERGY MATRIX (SPARSE MATRIX)
%       GLOB_V: POTENTIAL ENERGY MATRIX (SPARSE MATRIX)
%       GLOB_E: OVERLAP MATRIX (SPARSE MATRIX)
%       ENODES: NODES ON THE BOUNDING BOX (NEEDED FOR POST PROCESSING)
%       P: COORDINATES OF NODES (NNODES X 3)
%       T: TETRAHEDRAL ELEMENTS (NEL X NNEL)
%       VERT: VERTICES OF QUANTUM DOT (M X P,M VERTICES IN P DIMENSIONS)
%       V_NODES: POTENTIAL AT EACH NODE OTHER THAN OFFSET IN eV (NNODES X 1)
%-------------------------------------------------------------------------
%
%   Physical constants:
%       m_e: mass of electron in grams
%       h_bar: plancks constant in kilograms-Angstrom per second
%       q: measure of electron-volt in gram-centimeter squared per second
%       squared
%   Variables:
%       ndof: number of degrees of freedom (dof) per node (1 OR 4)
%       m_eff: effective mass of semiconductor in grams(1 X 2)
%       V_offset: conduction or valence band offset in eV (1 X 2)
%       nnel: number of nodes per element
%       nel: number of elements
%       nnodes: number of nodes
%       interface_nodes: nodes on interface separating the two materials
%       elem_size: size of element matrices
%       r,c,v: row & column indices of respective element matrix that point
%           to value v.
%       nd: local node indexing (1 X NNEL)
%       xcoord/ycoord: local coordinate indexing (1 X NNEL)
%       V: potential energy per node (1 X NNEL)
%       m: effective mass of element
%       J: Jacobian Matrix (2 X 2)
%       Elem_K: elemental kinetic matrix (EDOF X EDOF)
%       Elem_V: elemental potential matrix (EDOF X EDOF)
%       Elem_E: elemental overlap matrix (EDOF X EDOF)
%       index: system degree of freedom assigned to element node(1 X EDOF)
%   Functions:
%       surftri: Find surface triangles from mesh
%       inhull: Tests if a set of points is inside a convex hull
%       jacob_3D: The Jacobian for 2D mapping
%       Elem_matrix_K_3D: Computes the elemental kinetic matrix
%       Elem_matrix_V_3D: Computes the elemental potential matrix
%       Elem_matrix_E_3D: Computes the elemental overlap matrix
%       apply_interface_bc_3D: Applies the boundary conditions on the
%           interface between the two materials if dof > 1.
%       index_3D: Assigns the system dof to the element node
%       assemble: Builds up row & column indices
%       sparse: assembles global matrices by creating a sparse matrix
%
%ENTER USER INPUT HERE-----------------------------------------------------
ndof = 1;
m_e = 9.109*10^-28;
%(1) represents well and (2) represents barrier
m_eff(1)=m_e*0.009;
m_eff(2)=m_e*0.131;
V_offset(1)=0;
V_offset(2)=2.15;
%END USER INPUT-------------------------------------------------------------
%
h_bar = 1.054*10^-19;
```

```
q = 1.602*10^-12;
nnel = 4;
nel=length(t(:,1));
nnode=length(p(:,1));
interface_nodes = find(ismember(p,vert,'rows'));
%
%For bound state energies, which this program is solving for, boundary
%conditions require the wavefunction to fall off at regions approaching
%infinity.  In order to satisfy this, we bound the barrier region at
%"sufficient" distances.
%
enodes = unique(surftri(p,t));
%
elem_size = nnel*ndof*nnel*ndof;
r_K = zeros(nel,elem_size);r_V = zeros(nel,elem_size);r_E = zeros(nel,elem_size);
c_K = zeros(nel,elem_size);c_V = zeros(nel,elem_size);c_E = zeros(nel,elem_size);
v_K = zeros(nel,elem_size);v_V = zeros(nel,elem_size);v_E = zeros(nel,elem_size);
%
for iel=1:nel
    nd = t(iel,:);
    xcoord = p(nd,1);
    ycoord = p(nd,2);
    zcoord = p(nd,3);
    IN = inhull([xcoord ycoord zcoord],vert,[],0.01);
    if  (sum(IN) == nnel)
        V(1:nnel) = V_offset(1) + V_nodes(nd);
        m = m_eff(1);
    else
        V(1:nnel) = V_offset(2) + V_nodes(nd);
        m = m_eff(2);
    end
    J = jacob_3D(xcoord,ycoord,zcoord);
    %
    %Construct the elemental matrices by using functions Elem_matrix_K,
    %Elem_matrix_V, and Elem_matrix_E.
    elem_K = ((h_bar)^2/(2*m*q))*Elem_matrix_K_3D(J,ndof);
    elem_V = Elem_matrix_V_3D(J,V,ndof);
    elem_E = Elem_matrix_E_3D(J,ndof);
    %
    %Add reciprical mass condition at interface if there is more than 1
    %dof per node.
    if (ndof > 1 && m == m_eff(1))
        elem_K = apply_interface_bc_3D(elem_K,interface_nodes,nd,m_eff(2)/m);
        elem_V = apply_interface_bc_3D(elem_V,interface_nodes,nd,m_eff(2)/m);
        elem_E = apply_interface_bc_3D(elem_E,interface_nodes,nd,m_eff(2)/m);
    end
    %
    %Builds up row & column indices using assemble function
    %
    index = index_3D(nd,ndof);
    [r_K(iel,:) c_K(iel,:) v_K(iel,:)] = assemble(elem_K,index);
    [r_V(iel,:) c_V(iel,:) v_V(iel,:)] = assemble(elem_V,index);
    [r_E(iel,:) c_E(iel,:) v_E(iel,:)] = assemble(elem_E,index);
end
%
%Construct the global matrices using sparse and triples of
%rows/columns/values
%
glob_K = sparse(r_K,c_K,v_K,ndof*nnode,ndof*nnode);
```

```
glob_V = sparse(r_V,c_V,v_V,ndof*nnode,ndof*nnode);
glob_E = sparse(r_E,c_E,v_E,ndof*nnode,ndof*nnode);
%
end
```

B.4.1 Jacob_3D

```
function [ J ] = jacob_3D( x,y,z )
%Jacob_2D The Jacobian for the 3D mapping
%   The Jacobian is constant due to the linear transformation of local
%   coordinates to global coordinates.  Linear shape functions are used as
%   the coordinate transformation.
%
%   [J]=JACOB_3D(X,Y)
%      J: JACOBIAN MATRIX (3x3)
%      X: GLOBAL COORDINATES IN THE X DIRECTION(1x4)
%      Y: GLOBAL COORDINATES IN THE Y DIRECTION(1x4)
%      Z: GLOBAL COORDINATES IN THE Z DIRECTION(1X4)
%--------------------------------------------------------------------------
%
J(1,1) = x(2)-x(1);
J(1,2) = y(2)-y(1);
J(1,3) = z(2)-z(1);
J(2,1) = x(3)-x(1);
J(2,2) = y(3)-y(1);
J(2,3) = z(3)-z(1);
J(3,1) = x(4)-x(1);
J(3,2) = y(4)-y(1);
J(3,3) = z(4)-z(1);
end
```

B.4.2 Elem_matrix_K_3D

```
function [ elem_K ] = Elem_matrix_K_3D( J,ndof )
%Elem_matrix_K_3D Computes the elemental kinetic matrix
%   The elemental kinetic energy matrix is formed using either linear basis
%   functions or cubic basis functions.
%
%   [ELEM_K]=ELEM_MATRIX_K_3D(J,NDOF)
%      ELEM_K: ELEMENTAL KINETIC ENERGY MATRIX (4 X 4 OR 16 X 16)
%      J: JACOBIAN MATRIX(3 X 3)
%      NDOF: DEGREE OF FREEDOM PER NODE
%--------------------------------------------------------------------------
%The elemental kinetic energy matrix is constructed using the quadratic
%form as represented by the equation Trace(M_C*(JJ^T)^-1).  This form is
%convenient because the integrals are carried out once and represented in
%matrices M_C and vectors Mxx, Mxy, Mxz, Myz, Myy, Myz, Mzx, Mzy, and Mzz
%below.
%   Variables:
%       M_C: MATRIX REPRESENTING THE PRODUCT OF DERIVATIVES (3 X 3)
%       MXX: VECTOR REPRESENTING PRODUCT OF SHAPE FUNCTION DERIVATIVES WITH
%       RESPECT TO X (1 X 16 OR 1 X 256)
%       MXY: VECTOR REPRESENTING PRODUCT OF SHAPE FUNCTION DERIVATIVES WITH
%       RESPECT TO X AND Y (1 X 16 OR 1 X 256)
%       MXZ: VECTOR REPRESENTING PRODUCT OF SHAPE FUNCTION DERIVATIVES WITH
%       RESPECT TO X AND Z (1 X 16 OR 1 X 256)
%       MYX: VECTOR REPRESENTING PRODUCT OF SHAPE FUNCTION DERIVATIVES WITH
%       RESPECT TO Y AND X (1 X 16 OR 1 X 256)
%       MYY: VECTOR REPRESENTING PRODUCT OF SHAPE FUNCTION DERIVATIVES WITH
```

```
%       RESPECT TO Y (1 X 16 OR 1 X 256)
%       MYZ: VECTOR REPRESENTING PRODUCT OF SHAPE FUNCTION DERIVATIVES WITH
%       RESPECT TO Y AND Z (1 X 16 OR 1 X 256)
%       MZX: VECTOR REPRESENTING PRODUCT OF SHAPE FUNCTION DERIVATIVES WITH
%       RESPECT TO Z AND X (1 X 16 OR 1 X 256)
%       MZY: VECTOR REPRESENTING PRODUCT OF SHAPE FUNCTION DERIVATIVES WITH
%       RESPECT TO Z AND Y (1 X 16 OR 1 X 256)
%       MZZ: VECTOR REPRESENTING PRODUCT OF SHAPE FUNCTION DERIVATIVES WITH
%       RESPECT TO Z (1 X 16 OR 1 X 256)
%       TRANSLATE: Matrix that converts ELEM_K from local coordinates to
%       global coordinates.  This is only used when ndof does not equal 1.
%
%Linear shape functions are defined as;
%   N_1(x,y,z)=1-x-y-z;     0<=x,y,z<=1
%   N_2(x,y,z)=x;           0<=x,y,z<=1
%   N_3(x,y,z)=y;           0<=x,y,z<=1
%   N_4(x,y,z)=z;           0<=x,y,z<=1
%
%Cubic shape functions are defined as;
%   N_1(x,y,z)=1-N_5-N_9-N_13;                      0<=x,y,z<=1
%   N_2(x,y,z)=x-N_5-N_6-N_10-N_14;                 0<=x,y,z<=1
%   N_3(x,y,z)=y-N_7-N_9-N_11-N_15;                 0<=x,y,z<=1
%   N_4(x,y,z)=z-N_8-N_12-N_13-N_16                 0<=x,y,z<=1
%   N_5(x,y,z)=3*x^2-2*x^3;                         0<=x,y,z<=1
%   N_6(x,y,z)=-x^2+x^3;                            0<=x,y,z<=1
%   N_7(x,y,z)=x^2*y;                               0<=x,y,z<=1
%   N_8(x,y,z)=x^2*z;                               0<=x,y,z<=1
%   N_9(x,y,z)=3*y^2-2*y^3;                         0<=x,y,z<=1
%   N_10(x,y,z)=x*y^2;                              0<=x,y,z<=1
%   N_11(x,y,z)=-y^2+y^3;                           0<=x,y,z<=1
%   N_12(x,y,z)=y^2*z;                              0<=x,y,z<=1
%   N_13(x,y,z)=3*z^2-2*z^3;                        0<=x,y,z<=1
%   N_14(x,y,z)=x*z^2;                              0<=x,y,z<=1
%   N_15(x,y,z)=y*z^2;                              0<=x,y,z<=1
%   N_16(x,y,z)=-z^2+z^3;                           0<=x,y,z<=1
%-------------------------------------------------------------------------
%
K = (J*J')^-1;
if (ndof == 1)
    elem_K = zeros(4,4);
    Mxx = [1/6 -1/6 0 0  -1/6 1/6 0 0  0 0 0 0 0 0 0 0];
    Mxy = [1/6 0 -1/6 0  -1/6 0 1/6 0  0 0 0 0 0 0 0 0];
    Mxz = [1/6 0 0 -1/6  -1/6 0 0 1/6  0 0 0 0 0 0 0 0];
    Myx = [1/6 -1/6 0 0  0 0 0 0  -1/6 1/6 0 0 0 0 0 0];
    Myy = [1/6 0 -1/6 0  0 0 0 0  -1/6 0 1/6 0 0 0 0 0];
    Myz = [1/6 0 0 -1/6  0 0 0 0  -1/6 0 0 1/6 0 0 0 0];
    Mzx = [1/6 -1/6 0 0  0 0 0 0  0 0 0 0 -1/6 1/6 0 0];
    Mzy = [1/6 0 -1/6 0  0 0 0 0  0 0 0 0 -1/6 0 1/6 0];
    Mzz = [1/6 0 0 -1/6  0 0 0 0  0 0 0 0 -1/6 0 0 1/6];
    M = [Mxx ; Mxy ; Mxz ; Myx ; Myy ; Myz ; Mzx ; Mzy ; Mzz];
    M_C1 = reshape(M(:,1),3,3)'*K ; M_C2 = reshape(M(:,2),3,3)'*K;
    M_C3 = reshape(M(:,3),3,3)'*K ; M_C4 = reshape(M(:,4),3,3)'*K;
    M_C6 = reshape(M(:,6),3,3)'*K ; M_C7 = reshape(M(:,7),3,3)'*K;
    M_C8 = reshape(M(:,8),3,3)'*K ; M_C11 = reshape(M(:,11),3,3)'*K;
    M_C12 = reshape(M(:,12),3,3)'*K ; M_C16 = reshape(M(:,16),3,3)'*K;
    elem_K(1,1) = trace(M_C1); elem_K(1,2) = trace(M_C2);
    elem_K(1,3) = trace(M_C3); elem_K(1,4) = trace(M_C4);
    elem_K(2,2) = trace(M_C6); elem_K(2,3) = trace(M_C7);
    elem_K(2,4) = trace(M_C8); elem_K(3,3) = trace(M_C11);
```

```
    elem_K(3,4) = trace(M_C12); elem_K(4,4) = trace(M_C16);
    %
    elem_K = elem_K + elem_K' - diag(diag(elem_K));
    elem_K = det(J)*elem_K;
else
    elem_K = zeros(16,16);
    Mxx = [6/35, 1/105, 2/105, 2/105, -6/35, 1/28, -2/105, -2/105, 0,...
        -1/84, 0, 0, 0, -1/84, 0, 0  1/105, 1/45, 1/630, 1/630, -1/105,...
        -1/1260, -1/630, -1/630, 0, 1/420, 0, 0, 0, 1/420, 0, 0  2/105,...
        1/630, 1/315, 1/630, -2/105, 1/252, -1/315, -1/630, 0, -1/420, 0,...
        0, 0, -1/1260, 0, 0  2/105, 1/630, 1/630, 1/315, -2/105, 1/252,...
        -1/630, -1/315, 0, -1/1260, 0, 0, 0, -1/420, 0, 0  -6/35,...
        -1/105, -2/105, -2/105, 6/35, -1/28, 2/105, 2/105, 0, 1/84, 0, 0,...
        0, 1/84, 0, 0  1/28, -1/1260, 1/252, 1/252, -1/28, 1/105,...
        -1/252, -1/252, 0, -1/315, 0, 0, 0, -1/315, 0, 0  -2/105,...
        -1/630, -1/315, -1/630, 2/105, -1/252, 1/315, 1/630, 0, 1/420,...
        0, 0, 0, 1/1260, 0, 0  -2/105, -1/630, -1/630, -1/315, 2/105,...
        -1/252, 1/630, 1/315, 0, 1/1260, 0, 0, 0, 1/420, 0, 0  0, 0,...
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  -1/84, 1/420,...
        -1/420, -1/1260, 1/84, -1/315, 1/420, 1/1260, 0, 1/210, 0, 0,...
        0, 1/1260, 0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
        0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  0, 0,...
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  -1/84, 1/420,...
        -1/1260, -1/420, 1/84, -1/315, 1/1260, 1/420, 0, 1/1260, 0,...
        0, 0, 1/210, 0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
        0, 0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
    Mxy = [9/70, 2/105, -2/105, 1/84, 0, 0, -3/140, 0, -9/70, -2/105,...
        13/420, -1/84, 0, 0, -1/84, 0  -2/105, 1/630, 1/315, -1/315,...
        0, 0, -1/252, 0, 2/105, -1/630, -1/252, 1/315, 0, 0, 1/420, 0 ...
        2/105, 1/315, 1/630, 1/630, 0, 0, -1/420, 0, -2/105, -1/315, 1/252,...
        -1/630, 0, 0, -1/1260, 0  1/84, 1/630, -1/315, 1/630, 0, 0,...
        -1/420, 0, -1/84, -1/630, 1/315, -1/630, 0, 0, -1/420, 0 ...
        -9/70, -2/105, 2/105, -1/84, 0, 0, 3/140, 0, 9/70, 2/105, -13/420,...
        1/84, 0, 0, 1/84, 0  13/420, 1/252, -1/252, 1/315, 0, 0,...
        -1/420, 0, -13/420, -1/252, 1/140, -1/315, 0, 0, -1/315, 0 ...
        -2/105, -1/315, -1/630, -1/630, 0, 0, 1/420, 0, 2/105, 1/315,...
        -1/252, 1/630, 0, 0, 1/1260, 0  -1/84, -1/630, 1/315, -1/630,...
        0, 0, 1/420, 0, 1/84, 1/630, -1/315, 1/630, 0, 0, 1/420, 0 ...
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  -3/140, -1/420,...
        -1/252, -1/420, 0, 0, 1/1260, 0, 3/140, 1/420, -1/420, 1/420,...
        0, 0, 1/1260, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
        0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  0, 0, 0,...
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  -1/84, -1/1260, 1/420,...
        -1/420, 0, 0, 1/1260, 0, 1/84, 1/1260, -1/315, 1/420, 0, 0,...
        1/210, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ...
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
    Mxz = [9/70, 2/105, 1/84, -2/105, 0, 0, 0, -3/140, 0, 0, 0,...
        -1/84, -9/70, -2/105, -1/84, 13/420  -2/105, 1/630, -1/315,...
        1/315, 0, 0, 0, -1/252, 0, 0, 0, 1/420, 2/105, -1/630, 1/315,...
        -1/252  1/84, 1/630, 1/630, -1/315, 0, 0, 0, -1/420, 0, 0,...
        0, -1/420, -1/84, -1/630, -1/630, 1/315  2/105, 1/315, 1/630,...
        1/630, 0, 0, 0, -1/420, 0, 0, 0, -1/1260, -2/105, -1/315,...
        -1/630, 1/252  -9/70, -2/105, -1/84, 2/105, 0, 0, 0, 3/140,...
        0, 0, 0, 1/84, 9/70, 2/105, 1/84, -13/420  13/420, 1/252,...
        1/315, -1/252, 0, 0, 0, -1/420, 0, 0, 0, -1/315, -13/420,...
        -1/252, -1/315, 1/140  -1/84, -1/630, -1/630, 1/315, 0, 0,...
        0, 1/420, 0, 0, 0, 1/420, 1/84, 1/630, 1/630, -1/315  -2/105,...
        -1/315, -1/630, -1/630, 0, 0, 0, 1/420, 0, 0, 0, 1/1260, 2/105,...
        1/315, 1/630, -1/252  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
```

```
    0, 0, 0  -1/84, -1/1260, -1/420, 1/420, 0, 0, 0, 1/1260, 0,...
    0, 0, 1/210, 1/84, 1/1260, 1/420, -1/315  0, 0, 0, 0, 0, 0, 0,...
    0, 0, 0, 0, 0, 0, 0, 0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
    0, 0, 0, 0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
    0  -3/140, -1/420, -1/420, -1/252, 0, 0, 0, 1/1260, 0, 0, 0,...
    1/1260, 3/140, 1/420, 1/420, -1/420  0, 0, 0, 0, 0, 0, 0, 0,...
    0, 0, 0, 0, 0, 0, 0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
    0, 0, 0];
Myx = [9/70, -2/105, 2/105, 1/84, -9/70, 13/420, -2/105, -1/84, 0,...
    -3/140, 0, 0, 0, -1/84, 0, 0  2/105, 1/630, 1/315, 1/630,...
    -2/105, 1/252, -1/315, -1/630, 0, -1/420, 0, 0, 0, -1/1260, 0,...
    0  -2/105, 1/315, 1/630, -1/315, 2/105, -1/252, -1/630, 1/315,...
    0, -1/252, 0, 0, 0, 1/420, 0, 0  1/84, -1/315, 1/630, 1/630,...
    -1/84, 1/315, -1/630, -1/630, 0, -1/420, 0, 0, 0, -1/420, 0,...
    0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  0, 0,...
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  -3/140, -1/252,...
    -1/420, -1/420, 3/140, -1/420, 1/420, 1/420, 0, 1/1260, 0, 0,...
    0, 1/1260, 0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
    0, 0  -9/70, 2/105, -2/105, -1/84, 9/70, -13/420, 2/105, 1/84,...
    0, 3/140, 0, 0, 0, 1/84, 0, 0  -2/105, -1/630, -1/315, -1/630,...
    2/105, -1/252, 1/315, 1/630, 0, 1/420, 0, 0, 0, 1/1260, 0, 0 ...
    13/420, -1/252, 1/252, 1/315, -13/420, 1/140, -1/252, -1/315, 0,...
    -1/420, 0, 0, 0, -1/315, 0, 0  -1/84, 1/315, -1/630, -1/630,...
    1/84, -1/315, 1/630, 1/630, 0, 1/420, 0, 0, 0, 1/420, 0, 0 ...
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  0, 0, 0, 0, 0,...
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  -1/84, 1/420, -1/1260,...
    -1/420, 1/84, -1/315, 1/1260, 1/420, 0, 1/1260, 0, 0, 0, 1/210,...
    0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
Myy = [6/35, 2/105, 1/105, 2/105, 0, 0, -1/84, 0, -6/35, -2/105,...
    1/28, -2/105, 0, 0, -1/84, 0  2/105, 1/315, 1/630, 1/630, 0,...
    0, -1/420, 0, -2/105, -1/315, 1/252, -1/630, 0, 0, -1/1260, 0 ...
    1/105, 1/630, 1/45, 1/630, 0, 0, 1/420, 0, -1/105, -1/630, -1/1260,...
    -1/630, 0, 0, 1/420, 0  2/105, 1/630, 1/630, 1/315, 0, 0,...
    -1/1260, 0, -2/105, -1/630, 1/252, -1/315, 0, 0, -1/420, 0 ...
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  0, 0, 0, 0, 0,...
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  -1/84, -1/420, 1/420,...
    -1/1260, 0, 0, 1/210, 0, 1/84, 1/420, -1/315, 1/1260, 0, 0,...
    1/1260, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ...
    -6/35, -2/105, -1/105, -2/105, 0, 0, 1/84, 0, 6/35, 2/105, -1/28,...
    2/105, 0, 0, 1/84, 0  -2/105, -1/315, -1/630, -1/630, 0, 0,...
    1/420, 0, 2/105, 1/315, -1/252, 1/630, 0, 0, 1/1260, 0  1/28,...
    1/252, -1/1260, 1/252, 0, 0, -1/315, 0, -1/28, -1/252, 1/105,...
    -1/252, 0, 0, -1/315, 0  -2/105, -1/630, -1/630, -1/315, 0,...
    0, 1/1260, 0, 2/105, 1/630, -1/252, 1/315, 0, 0, 1/420, 0 ...
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  0, 0, 0, 0, 0,...
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  -1/84, -1/1260, 1/420,...
    -1/420, 0, 0, 1/1260, 0, 1/84, 1/1260, -1/315, 1/420, 0, 0,...
    1/210, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
Myz = [9/70, 1/84, 2/105, -2/105, 0, 0, 0, -1/84, 0, 0, 0, -3/140,...
    -9/70, -1/84, -2/105, 13/420  1/84, 1/630, 1/630, -1/315, 0,...
    0, 0, -1/420, 0, 0, 0, -1/420, -1/84, -1/630, -1/630, 1/315 ...
    -2/105, -1/315, 1/630, 1/315, 0, 0, 0, 1/420, 0, 0, 0, -1/252,...
    2/105, 1/315, -1/630, -1/252  2/105, 1/630, 1/315, 1/630,...
    0, 0, 0, -1/1260, 0, 0, 0, -1/420, -2/105, -1/630, -1/315,...
    1/252  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ...
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  -1/84, -1/420,...
    -1/1260, 1/420, 0, 0, 0, 1/210, 0, 0, 0, 1/1260, 1/84, 1/420,...
    1/1260, -1/315  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
    0  -9/70, -1/84, -2/105, 2/105, 0, 0, 0, 1/84, 0, 0, 0,...
```

```
    3/140, 9/70, 1/84, 2/105, -13/420  -1/84, -1/630, -1/630,...
    1/315, 0, 0, 0, 1/420, 0, 0, 0, 1/420, 1/84, 1/630, 1/630,...
    -1/315  13/420, 1/315, 1/252, -1/252, 0, 0, 0, -1/315, 0,...
    0, 0, -1/420, -13/420, -1/315, -1/252, 1/140  -2/105, -1/630,...
    -1/315, -1/630, 0, 0, 0, 1/1260, 0, 0, 0, 1/420, 2/105, 1/630,...
    1/315, -1/252  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
    0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  -3/140,...
    -1/420, -1/420, -1/252, 0, 0, 0, 1/1260, 0, 0, 0, 1/1260,...
    3/140, 1/420, 1/420, -1/420  0, 0, 0, 0, 0, 0, 0, 0, 0,...
    0, 0, 0, 0, 0, 0, 0];
Mzx = [9/70, -2/105, 1/84, 2/105, -9/70, 13/420, -1/84, -2/105, 0,...
    -1/84, 0, 0, 0, -3/140, 0, 0  2/105, 1/630, 1/630, 1/315,...
    -2/105, 1/252, -1/630, -1/315, 0, -1/1260, 0, 0, 0, -1/420,...
    0, 0  1/84, -1/315, 1/630, 1/630, -1/84, 1/315, -1/630,...
    -1/630, 0, -1/420, 0, 0, 0, -1/420, 0, 0  -2/105, 1/315,...
    -1/315, 1/630, 2/105, -1/252, 1/315, -1/630, 0, 1/420, 0, 0,...
    0, -1/252, 0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
    0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  0, 0,...
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  -3/140, -1/252,...
    -1/420, -1/420, 3/140, -1/420, 1/420, 1/420, 0, 1/1260, 0, 0,...
    0, 1/1260, 0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
    0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  0,...
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  -1/84, 1/420,...
    -1/420, -1/1260, 1/84, -1/315, 1/420, 1/1260, 0, 1/210, 0, 0,...
    0, 1/1260, 0, 0  -9/70, 2/105, -1/84, -2/105, 9/70, -13/420,...
    1/84, 2/105, 0, 1/84, 0, 0, 0, 3/140, 0, 0  -2/105, -1/630,...
    -1/630, -1/315, 2/105, -1/252, 1/630, 1/315, 0, 1/1260, 0, 0,...
    0, 1/420, 0, 0  -1/84, 1/315, -1/630, -1/630, 1/84, -1/315,...
    1/630, 1/630, 0, 1/420, 0, 0, 0, 1/420, 0, 0  13/420, -1/252,...
    1/315, 1/252, -13/420, 1/140, -1/315, -1/252, 0, -1/315, 0, 0,...
    0, -1/420, 0, 0];
Mzy = [9/70, 1/84, -2/105, 2/105, 0, 0, -1/84, 0, -9/70, -1/84,...
    13/420, -2/105, 0, 0, -3/140, 0  1/84, 1/630, -1/315, 1/630,...
    0, 0, -1/420, 0, -1/84, -1/630, 1/315, -1/630, 0, 0, -1/420,...
    0  2/105, 1/630, 1/630, 1/315, 0, 0, -1/1260, 0, -2/105,...
    -1/630, 1/252, -1/315, 0, 0, -1/420, 0  -2/105, -1/315,...
    1/315, 1/630, 0, 0, 1/420, 0, 2/105, 1/315, -1/252, -1/630,...
    0, 0, -1/252, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
    0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  0, 0,...
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  -1/84, -1/420,...
    1/420, -1/1260, 0, 0, 1/210, 0, 1/84, 1/420, -1/315, 1/1260,...
    0, 0, 1/1260, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
    0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  0,...
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  -3/140,...
    -1/420, -1/252, -1/420, 0, 0, 1/1260, 0, 3/140, 1/420,...
    -1/420, 1/420, 0, 0, 1/1260, 0  -9/70, -1/84, 2/105,...
    -2/105, 0, 0, 1/84, 0, 9/70, 1/84, -13/420, 2/105, 0, 0,...
    3/140, 0  -1/84, -1/630, 1/315, -1/630, 0, 0, 1/420, 0,...
    1/84, 1/630, -1/315, 1/630, 0, 0, 1/420, 0  -2/105,...
    -1/630, -1/630, -1/315, 0, 0, 1/1260, 0, 2/105, 1/630,...
    -1/252, 1/315, 0, 0, 1/420, 0  13/420, 1/315, -1/252,...
    1/252, 0, 0, -1/315, 0, -13/420, -1/315, 1/140, -1/252,...
    0, 0, -1/420, 0];
Mzz = [6/35, 2/105, 2/105, 1/105, 0, 0, 0, -1/84, 0, 0, 0,...
    -1/84, -6/35, -2/105, -2/105, 1/28  2/105, 1/315, 1/630,...
    1/630, 0, 0, 0, -1/420, 0, 0, 0, -1/1260, -2/105, -1/315,...
    -1/630, 1/252  2/105, 1/630, 1/315, 1/630, 0, 0, 0,...
    -1/1260, 0, 0, 0, -1/420, -2/105, -1/630, -1/315, 1/252 ...
    1/105, 1/630, 1/630, 1/45, 0, 0, 0, 1/420, 0, 0, 0, 1/420,...
```

```
    -1/105, -1/630, -1/630, -1/1260  0, 0, 0, 0, 0, 0, 0,...
    0, 0, 0, 0, 0, 0, 0, 0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0,...
    0, 0, 0, 0, 0, 0, 0  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
    0, 0, 0, 0, 0  -1/84, -1/420, -1/1260, 1/420, 0, 0, 0,...
    1/210, 0, 0, 0, 1/1260, 1/84, 1/420, 1/1260, -1/315  0, 0,...
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  0, 0, 0, 0, 0,...
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 0, 0, 0, 0, 0, 0, 0, 0, 0,...
    0, 0, 0, 0, 0, 0, 0, 0  -1/84, -1/1260, -1/420, 1/420, 0,...
    0, 0, 1/1260, 0, 0, 0, 1/210, 1/84, 1/1260, 1/420, -1/315 ...
    -6/35, -2/105, -2/105, -1/105, 0, 0, 0, 1/84, 0, 0, 0,...
    1/84, 6/35, 2/105, 2/105, -1/28  -2/105, -1/315, -1/630,...
    -1/630, 0, 0, 0, 1/420, 0, 0, 0, 1/1260, 2/105, 1/315, 1/630,...
    -1/252  -2/105, -1/630, -1/315, -1/630, 0, 0, 0, 1/1260, 0,...
    0, 0, 1/420, 2/105, 1/630, 1/315, -1/252  1/28, 1/252, 1/252,...
    -1/1260, 0, 0, 0, -1/315, 0, 0, 0, -1/315, -1/28, -1/252,...
    -1/252, 1/105];
M = [Mxx ; Mxy ; Mxz ; Myx ; Myy ; Myz ; Mzx ; Mzy ; Mzz];
M_C1 = reshape(M(:,1),3,3)'*K ; M_C2 = reshape(M(:,2),3,3)'*K;
M_C3 = reshape(M(:,3),3,3)'*K ; M_C4 = reshape(M(:,4),3,3)'*K;
M_C5 = reshape(M(:,5),3,3)'*K ; M_C6 = reshape(M(:,6),3,3)'*K;
M_C7 = reshape(M(:,7),3,3)'*K ; M_C8 = reshape(M(:,8),3,3)'*K;
M_C9 = reshape(M(:,9),3,3)'*K ; M_C10 = reshape(M(:,10),3,3)'*K;
M_C11 = reshape(M(:,11),3,3)'*K ; M_C12 = reshape(M(:,12),3,3)'*K;
M_C13 = reshape(M(:,13),3,3)'*K ; M_C14 = reshape(M(:,14),3,3)'*K;
M_C15 = reshape(M(:,15),3,3)'*K ; M_C16 = reshape(M(:,16),3,3)'*K;
M_C18 = reshape(M(:,18),3,3)'*K ; M_C19 = reshape(M(:,19),3,3)'*K;
M_C20 = reshape(M(:,20),3,3)'*K ; M_C21 = reshape(M(:,21),3,3)'*K;
M_C22 = reshape(M(:,22),3,3)'*K ; M_C23 = reshape(M(:,23),3,3)'*K;
M_C24 = reshape(M(:,24),3,3)'*K ; M_C25 = reshape(M(:,25),3,3)'*K;
M_C26 = reshape(M(:,26),3,3)'*K ; M_C27 = reshape(M(:,27),3,3)'*K;
M_C28 = reshape(M(:,28),3,3)'*K ; M_C29 = reshape(M(:,29),3,3)'*K;
M_C30 = reshape(M(:,30),3,3)'*K ; M_C31 = reshape(M(:,31),3,3)'*K;
M_C32 = reshape(M(:,32),3,3)'*K ; M_C35 = reshape(M(:,35),3,3)'*K;
M_C36 = reshape(M(:,36),3,3)'*K ; M_C37 = reshape(M(:,37),3,3)'*K;
M_C38 = reshape(M(:,38),3,3)'*K ; M_C39 = reshape(M(:,39),3,3)'*K;
M_C40 = reshape(M(:,40),3,3)'*K ; M_C41 = reshape(M(:,41),3,3)'*K;
M_C42 = reshape(M(:,42),3,3)'*K ; M_C43 = reshape(M(:,43),3,3)'*K;
M_C44 = reshape(M(:,44),3,3)'*K ; M_C45 = reshape(M(:,45),3,3)'*K;
M_C46 = reshape(M(:,46),3,3)'*K ; M_C47 = reshape(M(:,47),3,3)'*K;
M_C48 = reshape(M(:,48),3,3)'*K ; M_C52 = reshape(M(:,52),3,3)'*K;
M_C53 = reshape(M(:,53),3,3)'*K ; M_C54 = reshape(M(:,54),3,3)'*K;
M_C55 = reshape(M(:,55),3,3)'*K ; M_C56 = reshape(M(:,56),3,3)'*K;
M_C57 = reshape(M(:,57),3,3)'*K ; M_C58 = reshape(M(:,58),3,3)'*K;
M_C59 = reshape(M(:,59),3,3)'*K ; M_C60 = reshape(M(:,60),3,3)'*K;
M_C61 = reshape(M(:,61),3,3)'*K ; M_C62 = reshape(M(:,62),3,3)'*K;
M_C63 = reshape(M(:,63),3,3)'*K ; M_C64 = reshape(M(:,64),3,3)'*K;
M_C69 = reshape(M(:,69),3,3)'*K ; M_C70 = reshape(M(:,70),3,3)'*K;
M_C71 = reshape(M(:,71),3,3)'*K ; M_C72 = reshape(M(:,72),3,3)'*K;
M_C73 = reshape(M(:,73),3,3)'*K ; M_C74 = reshape(M(:,74),3,3)'*K;
M_C75 = reshape(M(:,75),3,3)'*K ; M_C76 = reshape(M(:,76),3,3)'*K;
M_C77 = reshape(M(:,77),3,3)'*K ; M_C78 = reshape(M(:,78),3,3)'*K;
M_C79 = reshape(M(:,79),3,3)'*K ; M_C80 = reshape(M(:,80),3,3)'*K;
M_C86 = reshape(M(:,86),3,3)'*K ; M_C87 = reshape(M(:,87),3,3)'*K;
M_C88 = reshape(M(:,88),3,3)'*K ; M_C89 = reshape(M(:,89),3,3)'*K;
M_C90 = reshape(M(:,90),3,3)'*K ; M_C91 = reshape(M(:,91),3,3)'*K;
M_C92 = reshape(M(:,92),3,3)'*K ; M_C93 = reshape(M(:,93),3,3)'*K;
M_C94 = reshape(M(:,94),3,3)'*K ; M_C95 = reshape(M(:,95),3,3)'*K;
M_C96 = reshape(M(:,96),3,3)'*K ; M_C103 = reshape(M(:,103),3,3)'*K;
M_C104 = reshape(M(:,104),3,3)'*K ; M_C105 = reshape(M(:,105),3,3)'*K;
```

```
M_C106 = reshape(M(:,106),3,3)'*K ; M_C107 = reshape(M(:,107),3,3)'*K;
M_C108 = reshape(M(:,108),3,3)'*K ; M_C109 = reshape(M(:,109),3,3)'*K;
M_C110 = reshape(M(:,110),3,3)'*K ; M_C111 = reshape(M(:,111),3,3)'*K;
M_C112 = reshape(M(:,112),3,3)'*K ; M_C120 = reshape(M(:,120),3,3)'*K;
M_C121 = reshape(M(:,121),3,3)'*K ; M_C122 = reshape(M(:,122),3,3)'*K;
M_C123 = reshape(M(:,123),3,3)'*K ; M_C124 = reshape(M(:,124),3,3)'*K;
M_C125 = reshape(M(:,125),3,3)'*K ; M_C126 = reshape(M(:,126),3,3)'*K;
M_C127 = reshape(M(:,127),3,3)'*K ; M_C128 = reshape(M(:,128),3,3)'*K;
M_C137 = reshape(M(:,137),3,3)'*K ; M_C138 = reshape(M(:,138),3,3)'*K;
M_C139 = reshape(M(:,139),3,3)'*K ; M_C140 = reshape(M(:,140),3,3)'*K;
M_C141 = reshape(M(:,141),3,3)'*K ; M_C142 = reshape(M(:,142),3,3)'*K;
M_C143 = reshape(M(:,143),3,3)'*K ; M_C144 = reshape(M(:,144),3,3)'*K;
M_C154 = reshape(M(:,154),3,3)'*K ; M_C155 = reshape(M(:,155),3,3)'*K;
M_C156 = reshape(M(:,156),3,3)'*K ; M_C157 = reshape(M(:,157),3,3)'*K;
M_C158 = reshape(M(:,158),3,3)'*K ; M_C159 = reshape(M(:,159),3,3)'*K;
M_C160 = reshape(M(:,160),3,3)'*K ; M_C171 = reshape(M(:,171),3,3)'*K;
M_C172 = reshape(M(:,172),3,3)'*K ; M_C173 = reshape(M(:,173),3,3)'*K;
M_C174 = reshape(M(:,174),3,3)'*K ; M_C175 = reshape(M(:,175),3,3)'*K;
M_C176 = reshape(M(:,176),3,3)'*K ; M_C188 = reshape(M(:,188),3,3)'*K;
M_C189 = reshape(M(:,189),3,3)'*K ; M_C190 = reshape(M(:,190),3,3)'*K;
M_C191 = reshape(M(:,191),3,3)'*K ; M_C192 = reshape(M(:,192),3,3)'*K;
M_C205 = reshape(M(:,205),3,3)'*K ; M_C206 = reshape(M(:,206),3,3)'*K;
M_C207 = reshape(M(:,207),3,3)'*K ; M_C208 = reshape(M(:,208),3,3)'*K;
M_C222 = reshape(M(:,222),3,3)'*K ; M_C223 = reshape(M(:,223),3,3)'*K;
M_C224 = reshape(M(:,224),3,3)'*K ; M_C239 = reshape(M(:,239),3,3)'*K;
M_C240 = reshape(M(:,240),3,3)'*K ; M_C256 = reshape(M(:,256),3,3)'*K;
%
elem_K(1,1) = trace(M_C1) ; elem_K(1,2) = trace(M_C2) ;
elem_K(1,3) = trace(M_C3) ; elem_K(1,4) = trace(M_C4) ;
elem_K(1,5) = trace(M_C5) ; elem_K(1,6) = trace(M_C6) ;
elem_K(1,7) = trace(M_C7) ; elem_K(1,8) = trace(M_C8) ;
elem_K(1,9) = trace(M_C9) ; elem_K(1,10) = trace(M_C10) ;
elem_K(1,11) = trace(M_C11) ; elem_K(1,12) = trace(M_C12) ;
elem_K(1,13) = trace(M_C13) ; elem_K(1,14) = trace(M_C14) ;
elem_K(1,15) = trace(M_C15) ; elem_K(1,16) = trace(M_C16) ;
elem_K(2,2) = trace(M_C18) ; elem_K(2,3) = trace(M_C19) ;
elem_K(2,4) = trace(M_C20) ; elem_K(2,5) = trace(M_C21) ;
elem_K(2,6) = trace(M_C22) ; elem_K(2,7) = trace(M_C23) ;
elem_K(2,8) = trace(M_C24) ; elem_K(2,9) = trace(M_C25) ;
elem_K(2,10) = trace(M_C26) ; elem_K(2,11) = trace(M_C27) ;
elem_K(2,12) = trace(M_C28) ; elem_K(2,13) = trace(M_C29) ;
elem_K(2,14) = trace(M_C30) ; elem_K(2,15) = trace(M_C31) ;
elem_K(2,16) = trace(M_C32) ; elem_K(3,3) = trace(M_C35) ;
elem_K(3,4) = trace(M_C36) ; elem_K(3,5) = trace(M_C37) ;
elem_K(3,6) = trace(M_C38) ; elem_K(3,7) = trace(M_C39) ;
elem_K(3,8) = trace(M_C40) ; elem_K(3,9) = trace(M_C41) ;
elem_K(3,10) = trace(M_C42) ; elem_K(3,11) = trace(M_C43) ;
elem_K(3,12) = trace(M_C44) ; elem_K(3,13) = trace(M_C45) ;
elem_K(3,14) = trace(M_C46) ; elem_K(3,15) = trace(M_C47) ;
elem_K(3,16) = trace(M_C48) ; elem_K(4,4) = trace(M_C52) ;
elem_K(4,5) = trace(M_C53) ; elem_K(4,6) = trace(M_C54) ;
elem_K(4,7) = trace(M_C55) ; elem_K(4,8) = trace(M_C56) ;
elem_K(4,9) = trace(M_C57) ; elem_K(4,10) = trace(M_C58) ;
elem_K(4,11) = trace(M_C59) ; elem_K(4,12) = trace(M_C60) ;
elem_K(4,13) = trace(M_C61) ; elem_K(4,14) = trace(M_C62) ;
elem_K(4,15) = trace(M_C63) ; elem_K(4,16) = trace(M_C64) ;
elem_K(5,5) = trace(M_C69) ; elem_K(5,6) = trace(M_C70) ;
elem_K(5,7) = trace(M_C71) ; elem_K(5,8) = trace(M_C72) ;
elem_K(5,9) = trace(M_C73) ; elem_K(5,10) = trace(M_C74) ;
```

```
    elem_K(5,11) = trace(M_C75) ; elem_K(5,12) = trace(M_C76) ;
    elem_K(5,13) = trace(M_C77) ; elem_K(5,14) = trace(M_C78) ;
    elem_K(5,15) = trace(M_C79) ; elem_K(5,16) = trace(M_C80) ;
    elem_K(6,6) = trace(M_C86) ; elem_K(6,7) = trace(M_C87) ;
    elem_K(6,8) = trace(M_C88) ; elem_K(6,9) = trace(M_C89) ;
    elem_K(6,10) = trace(M_C90) ; elem_K(6,11) = trace(M_C91) ;
    elem_K(6,12) = trace(M_C92) ; elem_K(6,13) = trace(M_C93) ;
    elem_K(6,14) = trace(M_C94) ; elem_K(6,15) = trace(M_C95) ;
    elem_K(6,16) = trace(M_C96) ; elem_K(7,7) = trace(M_C103);
    elem_K(7,8) = trace(M_C104) ; elem_K(7,9) = trace(M_C105) ;
    elem_K(7,10) = trace(M_C106) ; elem_K(7,11) = trace(M_C107) ;
    elem_K(7,12) = trace(M_C108) ; elem_K(7,13) = trace(M_C109) ;
    elem_K(7,14) = trace(M_C110) ; elem_K(7,15) = trace(M_C111) ;
    elem_K(7,16) = trace(M_C112) ; elem_K(8,8) = trace(M_C120) ;
    elem_K(8,9) = trace(M_C121) ; elem_K(8,10) = trace(M_C122) ;
    elem_K(8,11) = trace(M_C123) ; elem_K(8,12) = trace(M_C124) ;
    elem_K(8,13) = trace(M_C125) ; elem_K(8,14) = trace(M_C126) ;
    elem_K(8,15) = trace(M_C127) ; elem_K(8,16) = trace(M_C128) ;
    elem_K(9,9) = trace(M_C137) ; elem_K(9,10) = trace(M_C138) ;
    elem_K(9,11) = trace(M_C139) ; elem_K(9,12) = trace(M_C140) ;
    elem_K(9,13) = trace(M_C141) ; elem_K(9,14) = trace(M_C142) ;
    elem_K(9,15) = trace(M_C143) ; elem_K(9,16) = trace(M_C144) ;
    elem_K(10,10) = trace(M_C154) ; elem_K(10,11) = trace(M_C155) ;
    elem_K(10,12) = trace(M_C156) ; elem_K(10,13) = trace(M_C157) ;
    elem_K(10,14) = trace(M_C158) ; elem_K(10,15) = trace(M_C159) ;
    elem_K(10,16) = trace(M_C160) ; elem_K(11,11) = trace(M_C171) ;
    elem_K(11,12) = trace(M_C172) ; elem_K(11,13) = trace(M_C173) ;
    elem_K(11,14) = trace(M_C174) ; elem_K(11,15) = trace(M_C175) ;
    elem_K(11,16) = trace(M_C176) ; elem_K(12,12) = trace(M_C188) ;
    elem_K(12,13) = trace(M_C189) ; elem_K(12,14) = trace(M_C190) ;
    elem_K(12,15) = trace(M_C191) ; elem_K(12,16) = trace(M_C192) ;
    elem_K(13,13) = trace(M_C205) ; elem_K(13,14) = trace(M_C206) ;
    elem_K(13,15) = trace(M_C207) ; elem_K(13,16) = trace(M_C208) ;
    elem_K(14,14) = trace(M_C222) ; elem_K(14,15) = trace(M_C223) ;
    elem_K(14,16) = trace(M_C224) ; elem_K(15,15) = trace(M_C239) ;
    elem_K(15,16) = trace(M_C240) ; elem_K(16,16) = trace(M_C256) ;
    %
    translate = zeros(16,16);
    translate(1,1) = 1;
    translate(2:4,2:4) = J;
    translate(5,5) = 1;
    translate(6:8,6:8) = J;
    translate(9,9) = 1;
    translate(10:12,10:12) = J;
    translate(13,13) = 1;
    translate(14:16,14:16) = J;
    elem_K = elem_K + elem_K' - diag(diag(elem_K));
    elem_K = det(J)*(translate'*elem_K*translate);
end
```

B.4.3 Elem_matrix_V_3D

```
function [ elem_V ] = Elem_matrix_V_3D( J,V,ndof )
%Elem_matrix_V_3D Computes the elemental potential matrix
%   The elemental potential energy matrix is formed using either linear
%   basis functions or cubic basis functions.
%
%   [ELEM_V]=ELEM_MATRIX_V_3D(J,V,NDOF)
%      ELEM_V: ELEMENTAL POTENTIAL ENERGY MATRIX (4 X 4 OR 16 X 16)
```

```
%      J: JACOBIAN MATRIX (3 X 3)
%      V: VECTOR CONTAINING POTENTIAL AT NODE (1 X 4)
%      NDOF: DEGREE OF FREEDOM PER NODE
%-------------------------------------------------------------------------
%
%   Variables:
%       Translate: Matrix that converts ELEM_V from local coordinates to
%       global coordinates.  This is only used when ndof does not equal 1.
%
%Linear shape functions are defined as;
%   N_1(x,y)=1-x-y-z;     0<=x,y,z<=1
%   N_2(x,y)=x;           0<=x,y,z<=1
%   N_3(x,y)=y;           0<=x,y,z<=1
%   N_4(x,y)=z;           0<=x,y,z<=1
%
%Cubic shape functions are defined as;
%   N_1(x,y,z)=1-N_5-N_9-N_13;                      0<=x,y,z<=1
%   N_2(x,y,z)=x-N_5-N_6-N_10-N_14;                 0<=x,y,z<=1
%   N_3(x,y,z)=y-N_7-N_9-N_11-N_15;                 0<=x,y,z<=1
%   N_4(x,y,z)=z-N_8-N_12-N_13-N_16                 0<=x,y,z<=1
%   N_5(x,y,z)=3*x^2-2*x^3;                         0<=x,y,z<=1
%   N_6(x,y,z)=-x^2+x^3;                            0<=x,y,z<=1
%   N_7(x,y,z)=x^2*y;                               0<=x,y,z<=1
%   N_8(x,y,z)=x^2*z;                               0<=x,y,z<=1
%   N_9(x,y,z)=3*y^2-2*y^3;                         0<=x,y,z<=1
%   N_10(x,y,z)=x*y^2;                              0<=x,y,z<=1
%   N_11(x,y,z)=-y^2+y^3;                           0<=x,y,z<=1
%   N_12(x,y,z)=y^2*z;                              0<=x,y,z<=1
%   N_13(x,y,z)=3*z^2-2*z^3;                        0<=x,y,z<=1
%   N_14(x,y,z)=x*z^2;                              0<=x,y,z<=1
%   N_15(x,y,z)=y*z^2;                              0<=x,y,z<=1
%   N_16(x,y,z)=-z^2+z^3;                           0<=x,y,z<=1
%-------------------------------------------------------------------------
%
if (ndof == 1)
    elem_V = 1/720*[6*V(1)+2*V(2)+2*V(3)+2*V(4) 2*V(1)+2*V(2)+1*V(3)+1*V(4)...
            2*V(1)+1*V(2)+2*V(3)+1*V(4) 2*V(1)+1*V(2)+1*V(3)+2*V(4) ; ...
            2*V(1)+2*V(2)+1*V(3)+1*V(4) 2*V(1)+6*V(2)+2*V(3)+2*V(4)...
            1*V(1)+2*V(2)+2*V(3)+1*V(4) 1*V(1)+1*V(2)+2*V(3)+2*V(4) ; ...
            2*V(1)+1*V(2)+2*V(3)+1*V(4) 1*V(1)+2*V(2)+2*V(3)+1*V(4)...
            2*V(1)+2*V(2)+6*V(3)+2*V(4) 1*V(1)+1*V(2)+2*V(3)+2*V(4) ; ...
            2*V(1)+1*V(2)+1*V(3)+2*V(4) 1*V(1)+2*V(2)+1*V(3)+2*V(4)...
            1*V(1)+1*V(2)+2*V(3)+2*V(4) 2*V(1)+2*V(2)+2*V(3)+6*V(4)];

    elem_V = det(J)*elem_V;
else
    elem_V = [(1/150)*V(3)+(1/150)*V(4)+(1/150)*V(2)+(11/700)*V(1),...
        (7/7200)*V(3)+(7/7200)*V(4)+(1/756)*V(2)+(143/75600)*V(1),...
        (1/756)*V(3)+(7/7200)*V(4)+(7/7200)*V(2)+(143/75600)*V(1),...
        (7/7200)*V(3)+(1/756)*V(4)+(7/7200)*V(2)+(143/75600)*V(1),...
        (1/560)*V(3)+(1/560)*V(4)+(17/4200)*V(2)+(17/6300)*V(1),...
        -(1/2016)*V(3)-(1/2016)*V(4)-(13/12600)*V(2)-(19/25200)*V(1),...
        (1/4725)*V(3)+(1/8400)*V(4)+(1/3360)*V(2)+(1/6048)*V(1),...
        (1/8400)*V(3)+(1/4725)*V(4)+(1/3360)*V(2)+(1/6048)*V(1),...
        (17/4200)*V(3)+(1/560)*V(4)+(1/560)*V(2)+(17/6300)*V(1),...
        (1/3360)*V(3)+(1/8400)*V(4)+(1/4725)*V(2)+(1/6048)*V(1),...
        -(13/12600)*V(3)-(1/2016)*V(4)-(1/2016)*V(2)-(19/25200)*V(1),...
        (1/3360)*V(3)+(1/4725)*V(4)+(1/8400)*V(2)+(1/6048)*V(1),...
        (1/560)*V(3)+(17/4200)*V(4)+(1/560)*V(2)+(17/6300)*V(1),...
```

```
(1/8400)*V(3)+(1/3360)*V(4)+(1/4725)*V(2)+(1/6048)*V(1),...
(1/4725)*V(3)+(1/3360)*V(4)+(1/8400)*V(2)+(1/6048)*V(1),...
-(1/2016)*V(3)-(13/12600)*V(4)-(1/2016)*V(2)-(19/25200)*V(1) ;...
(7/7200)*V(3)+(7/7200)*V(4)+(1/756)*V(2)+(143/75600)*V(1),...
(1/5400)*V(3)+(1/5400)*V(4)+(11/37800)*V(2)+(1/3240)*V(1),...
(23/113400)*V(3)+(23/151200)*V(4)+(23/113400)*V(2)+(107/453600)*V(1),...
(23/151200)*V(3)+(23/113400)*V(4)+(23/113400)*V(2)+(107/453600)*V(1),...
(11/25200)*V(3)+(11/25200)*V(4)+(1/945)*V(2)+(11/18900)*V(1),...
-(1/8400)*V(3)-(1/8400)*V(4)-(1/3780)*V(2)-(1/6300)*V(1),...
(1/18900)*V(3)+(1/33600)*V(4)+(1/12600)*V(2)+(11/302400)*V(1),...
(1/33600)*V(3)+(1/18900)*V(4)+(1/12600)*V(2)+(11/302400)*V(1),...
(17/25200)*V(3)+(1/3360)*V(4)+(1/2520)*V(2)+(53/151200)*V(1),...
(1/15120)*V(3)+(1/37800)*V(4)+(1/18900)*V(2)+(1/32400)*V(1),...
-(13/75600)*V(3)-(1/12096)*V(4)-(1/9072)*V(2)-(89/907200)*V(1),...
(1/20160)*V(3)+(1/28350)*V(4)+(1/37800)*V(2)+(19/907200)*V(1),...
(1/3360)*V(3)+(17/25200)*V(4)+(1/2520)*V(2)+(53/151200)*V(1),...
(1/37800)*V(3)+(1/15120)*V(4)+(1/18900)*V(2)+(1/32400)*V(1),...
(1/28350)*V(3)+(1/20160)*V(4)+(1/37800)*V(2)+(19/907200)*V(1),...
-(1/12096)*V(3)-(13/75600)*V(4)-(1/9072)*V(2)-(89/907200)*V(1) ;...
(1/756)*V(3)+(7/7200)*V(4)+(7/7200)*V(2)+(143/75600)*V(1),...
(23/113400)*V(3)+(23/151200)*V(4)+(23/113400)*V(2)+(107/453600)*V(1),...
(11/37800)*V(3)+(1/5400)*V(4)+(1/5400)*V(2)+(1/3240)*V(1),...
(23/113400)*V(3)+(23/113400)*V(4)+(23/151200)*V(2)+(107/453600)*V(1),...
(1/2520)*V(3)+(1/3360)*V(4)+(17/25200)*V(2)+(53/151200)*V(1),...
-(1/9072)*V(3)-(1/12096)*V(4)-(13/75600)*V(2)-(89/907200)*V(1),...
(1/18900)*V(3)+(1/37800)*V(4)+(1/15120)*V(2)+(1/32400)*V(1),...
(1/37800)*V(3)+(1/28350)*V(4)+(1/20160)*V(2)+(19/907200)*V(1),...
(1/945)*V(3)+(11/25200)*V(4)+(11/25200)*V(2)+(11/18900)*V(1),...
(1/12600)*V(3)+(1/33600)*V(4)+(1/18900)*V(2)+(11/302400)*V(1),...
-(1/3780)*V(3)-(1/8400)*V(4)-(1/8400)*V(2)-(1/6300)*V(1),...
(1/12600)*V(3)+(1/18900)*V(4)+(1/33600)*V(2)+(11/302400)*V(1),...
(1/2520)*V(3)+(17/25200)*V(4)+(1/3360)*V(2)+(53/151200)*V(1),...
(1/37800)*V(3)+(1/20160)*V(4)+(1/28350)*V(2)+(19/907200)*V(1),...
(1/18900)*V(3)+(1/15120)*V(4)+(1/37800)*V(2)+(1/32400)*V(1),...
-(1/9072)*V(3)-(13/75600)*V(4)-(1/12096)*V(2)-(89/907200)*V(1) ;...
(7/7200)*V(3)+(1/756)*V(4)+(7/7200)*V(2)+(143/75600)*V(1),...
(23/151200)*V(3)+(23/113400)*V(4)+(23/113400)*V(2)+(107/453600)*V(1),...
(23/113400)*V(3)+(23/113400)*V(4)+(23/151200)*V(2)+(107/453600)*V(1),...
(1/5400)*V(3)+(11/37800)*V(4)+(1/5400)*V(2)+(1/3240)*V(1),...
(1/3360)*V(3)+(1/2520)*V(4)+(17/25200)*V(2)+(53/151200)*V(1),...
-(1/12096)*V(3)-(1/9072)*V(4)-(13/75600)*V(2)-(89/907200)*V(1),...
(1/28350)*V(3)+(1/37800)*V(4)+(1/20160)*V(2)+(19/907200)*V(1),...
(1/37800)*V(3)+(1/18900)*V(4)+(1/15120)*V(2)+(1/32400)*V(1),...
(17/25200)*V(3)+(1/2520)*V(4)+(1/3360)*V(2)+(53/151200)*V(1),...
(1/20160)*V(3)+(1/37800)*V(4)+(1/28350)*V(2)+(19/907200)*V(1),...
-(13/75600)*V(3)-(1/9072)*V(4)-(1/12096)*V(2)-(89/907200)*V(1),...
(1/15120)*V(3)+(1/18900)*V(4)+(1/37800)*V(2)+(1/32400)*V(1),...
(11/25200)*V(3)+(1/945)*V(4)+(11/25200)*V(2)+(11/18900)*V(1),...
(1/33600)*V(3)+(1/12600)*V(4)+(1/18900)*V(2)+(11/302400)*V(1),...
(1/18900)*V(3)+(1/12600)*V(4)+(1/33600)*V(2)+(11/302400)*V(1),...
-(1/8400)*V(3)-(1/3780)*V(4)-(1/8400)*V(2)-(1/6300)*V(1) ;...
(1/560)*V(3)+(1/560)*V(4)+(17/4200)*V(2)+(17/6300)*V(1),...
(11/25200)*V(3)+(11/25200)*V(4)+(1/945)*V(2)+(11/18900)*V(1),...
(1/2520)*V(3)+(1/3360)*V(4)+(17/25200)*V(2)+(53/151200)*V(1),...
(1/3360)*V(3)+(1/2520)*V(4)+(17/25200)*V(2)+(53/151200)*V(1),...
(11/5040)*V(3)+(11/5040)*V(4)+(43/5040)*V(2)+(11/5040)*V(1),...
-(1/1890)*V(3)-(1/1890)*V(4)-(1/560)*V(2)-(1/1890)*V(1),...
(1/3780)*V(3)+(1/7560)*V(4)+(1/1680)*V(2)+(1/7560)*V(1),...
(1/7560)*V(3)+(1/3780)*V(4)+(1/1680)*V(2)+(1/7560)*V(1),...
```

```
(73/50400)*V(3)+(3/5600)*V(4)+(73/50400)*V(2)+(3/5600)*V(1),...
(11/50400)*V(3)+(11/151200)*V(4)+(1/3780)*V(2)+(11/151200)*V(1),...
-(19/50400)*V(3)-(23/151200)*V(4)-(31/75600)*V(2)-(23/151200)*V(1),...
(1/12600)*V(3)+(1/18900)*V(4)+(11/151200)*V(2)+(1/37800)*V(1),...
(3/5600)*V(3)+(73/50400)*V(4)+(73/50400)*V(2)+(3/5600)*V(1),...
(11/151200)*V(3)+(11/50400)*V(4)+(1/3780)*V(2)+(11/151200)*V(1),...
(1/18900)*V(3)+(1/12600)*V(4)+(11/151200)*V(2)+(1/37800)*V(1),...
-(23/151200)*V(3)-(19/50400)*V(4)-(31/75600)*V(2)-(23/151200)*V(1) ;...
-(1/2016)*V(3)-(1/2016)*V(4)-(13/12600)*V(2)-(19/25200)*V(1),...
-(1/8400)*V(3)-(1/8400)*V(4)-(1/3780)*V(2)-(1/6300)*V(1),...
-(1/9072)*V(3)-(1/12096)*V(4)-(13/75600)*V(2)-(89/907200)*V(1),...
-(1/12096)*V(3)-(1/9072)*V(4)-(13/75600)*V(2)-(89/907200)*V(1),...
-(1/1890)*V(3)-(1/1890)*V(4)-(1/560)*V(2)-(1/1890)*V(1),...
(1/7560)*V(3)+(1/7560)*V(4)+(1/2520)*V(2)+(1/7560)*V(1),...
-(1/15120)*V(3)-(1/30240)*V(4)-(1/7560)*V(2)-(1/30240)*V(1),...
-(1/30240)*V(3)-(1/15120)*V(4)-(1/7560)*V(2)-(1/30240)*V(1),...
-(31/75600)*V(3)-(23/151200)*V(4)-(19/50400)*V(2)-(23/151200)*V(1),...
-(1/16800)*V(3)-(1/50400)*V(4)-(1/15120)*V(2)-(1/50400)*V(1),...
(1/9450)*V(3)+(13/302400)*V(4)+(1/9450)*V(2)+(13/302400)*V(1),...
-(1/43200)*V(3)-(1/64800)*V(4)-(1/50400)*V(2)-(1/129600)*V(1),...
-(23/151200)*V(3)-(31/75600)*V(4)-(19/50400)*V(2)-(23/151200)*V(1),...
-(1/50400)*V(3)-(1/16800)*V(4)-(1/15120)*V(2)-(1/50400)*V(1),...
-(1/64800)*V(3)-(1/43200)*V(4)-(1/50400)*V(2)-(1/129600)*V(1),...
(13/302400)*V(3)+(1/9450)*V(4)+(1/9450)*V(2)+(13/302400)*V(1) ;...
(1/4725)*V(3)+(1/8400)*V(4)+(1/3360)*V(2)+(1/6048)*V(1),...
(1/18900)*V(3)+(1/33600)*V(4)+(1/12600)*V(2)+(11/302400)*V(1),...
(1/18900)*V(3)+(1/37800)*V(4)+(1/15120)*V(2)+(1/32400)*V(1),...
(1/28350)*V(3)+(1/37800)*V(4)+(1/20160)*V(2)+(19/907200)*V(1),...
(1/3780)*V(3)+(1/7560)*V(4)+(1/1680)*V(2)+(1/7560)*V(1),...
-(1/15120)*V(3)-(1/30240)*V(4)-(1/7560)*V(2)-(1/30240)*V(1),...
(1/25200)*V(3)+(1/75600)*V(4)+(1/15120)*V(2)+(1/75600)*V(1),...
(1/75600)*V(3)+(1/75600)*V(4)+(1/30240)*V(2)+(1/151200)*V(1),...
(1/3780)*V(3)+(11/151200)*V(4)+(11/50400)*V(2)+(11/151200)*V(1),...
(1/25200)*V(3)+(1/100800)*V(4)+(1/25200)*V(2)+(1/100800)*V(1),...
-(1/15120)*V(3)-(1/50400)*V(4)-(1/16800)*V(2)-(1/50400)*V(1),...
(1/75600)*V(3)+(1/151200)*V(4)+(1/100800)*V(2)+(1/302400)*V(1),...
(1/18900)*V(3)+(11/151200)*V(4)+(1/12600)*V(2)+(1/37800)*V(1),...
(1/151200)*V(3)+(1/100800)*V(4)+(1/75600)*V(2)+(1/302400)*V(1),...
(1/151200)*V(3)+(1/151200)*V(4)+(1/151200)*V(2)+(1/453600)*V(1),...
-(1/64800)*V(3)-(1/50400)*V(4)-(1/43200)*V(2)-(1/129600)*V(1) ;...
(1/8400)*V(3)+(1/4725)*V(4)+(1/3360)*V(2)+(1/6048)*V(1),...
(1/33600)*V(3)+(1/18900)*V(4)+(1/12600)*V(2)+(11/302400)*V(1),...
(1/37800)*V(3)+(1/28350)*V(4)+(1/20160)*V(2)+(19/907200)*V(1),...
(1/37800)*V(3)+(1/18900)*V(4)+(1/15120)*V(2)+(1/32400)*V(1),...
(1/7560)*V(3)+(1/3780)*V(4)+(1/1680)*V(2)+(1/7560)*V(1),...
-(1/30240)*V(3)-(1/15120)*V(4)-(1/7560)*V(2)-(1/30240)*V(1),...
(1/75600)*V(3)+(1/75600)*V(4)+(1/30240)*V(2)+(1/151200)*V(1),...
(1/75600)*V(3)+(1/25200)*V(4)+(1/15120)*V(2)+(1/75600)*V(1),...
(11/151200)*V(3)+(1/18900)*V(4)+(1/12600)*V(2)+(1/37800)*V(1),...
(1/100800)*V(3)+(1/151200)*V(4)+(1/75600)*V(2)+(1/302400)*V(1),...
-(1/50400)*V(3)-(1/64800)*V(4)-(1/43200)*V(2)-(1/129600)*V(1),...
(1/151200)*V(3)+(1/151200)*V(4)+(1/151200)*V(2)+(1/453600)*V(1),...
(11/151200)*V(3)+(1/3780)*V(4)+(11/50400)*V(2)+(11/151200)*V(1),...
(1/100800)*V(3)+(1/25200)*V(4)+(1/25200)*V(2)+(1/100800)*V(1),...
(1/151200)*V(3)+(1/75600)*V(4)+(1/100800)*V(2)+(1/302400)*V(1),...
-(1/50400)*V(3)-(1/15120)*V(4)-(1/16800)*V(2)-(1/50400)*V(1) ;...
(17/4200)*V(3)+(1/560)*V(4)+(1/560)*V(2)+(17/6300)*V(1),...
(17/25200)*V(3)+(1/3360)*V(4)+(1/2520)*V(2)+(53/151200)*V(1),...
(1/945)*V(3)+(11/25200)*V(4)+(11/25200)*V(2)+(11/18900)*V(1),...
```

```
(17/25200)*V(3)+(1/2520)*V(4)+(1/3360)*V(2)+(53/151200)*V(1),...
(73/50400)*V(3)+(3/5600)*V(4)+(73/50400)*V(2)+(3/5600)*V(1),...
-(31/75600)*V(3)-(23/151200)*V(4)-(19/50400)*V(2)-(23/151200)*V(1),...
(1/3780)*V(3)+(11/151200)*V(4)+(11/50400)*V(2)+(11/151200)*V(1),...
(11/151200)*V(3)+(1/18900)*V(4)+(1/12600)*V(2)+(1/37800)*V(1),...
(43/5040)*V(3)+(11/5040)*V(4)+(11/5040)*V(2)+(11/5040)*V(1),...
(1/1680)*V(3)+(1/7560)*V(4)+(1/3780)*V(2)+(1/7560)*V(1),...
-(1/560)*V(3)-(1/1890)*V(4)-(1/1890)*V(2)-(1/1890)*V(1),...
(1/1680)*V(3)+(1/3780)*V(4)+(1/7560)*V(2)+(1/7560)*V(1),...
(73/50400)*V(3)+(73/50400)*V(4)+(3/5600)*V(2)+(3/5600)*V(1),...
(11/151200)*V(3)+(1/12600)*V(4)+(1/18900)*V(2)+(1/37800)*V(1),...
(1/3780)*V(3)+(11/50400)*V(4)+(11/151200)*V(2)+(11/151200)*V(1),...
-(31/75600)*V(3)-(19/50400)*V(4)-(23/151200)*V(2)-(23/151200)*V(1) ;...
(1/3360)*V(3)+(1/8400)*V(4)+(1/4725)*V(2)+(1/6048)*V(1),...
(1/15120)*V(3)+(1/37800)*V(4)+(1/18900)*V(2)+(1/32400)*V(1),...
(1/12600)*V(3)+(1/33600)*V(4)+(1/18900)*V(2)+(11/302400)*V(1),...
(1/20160)*V(3)+(1/37800)*V(4)+(1/28350)*V(2)+(19/907200)*V(1),...
(11/50400)*V(3)+(11/151200)*V(4)+(1/3780)*V(2)+(11/151200)*V(1),...
-(1/16800)*V(3)-(1/50400)*V(4)-(1/15120)*V(2)-(1/50400)*V(1),...
(1/25200)*V(3)+(1/100800)*V(4)+(1/25200)*V(2)+(1/100800)*V(1),...
(1/100800)*V(3)+(1/151200)*V(4)+(1/75600)*V(2)+(1/302400)*V(1),...
(1/1680)*V(3)+(1/7560)*V(4)+(1/3780)*V(2)+(1/7560)*V(1),...
(1/15120)*V(3)+(1/75600)*V(4)+(1/25200)*V(2)+(1/75600)*V(1),...
-(1/7560)*V(3)-(1/30240)*V(4)-(1/15120)*V(2)-(1/30240)*V(1),...
(1/30240)*V(3)+(1/75600)*V(4)+(1/75600)*V(2)+(1/151200)*V(1),...
(1/12600)*V(3)+(11/151200)*V(4)+(1/18900)*V(2)+(1/37800)*V(1),...
(1/151200)*V(3)+(1/151200)*V(4)+(1/151200)*V(2)+(1/453600)*V(1),...
(1/75600)*V(3)+(1/100800)*V(4)+(1/151200)*V(2)+(1/302400)*V(1),...
-(1/43200)*V(3)-(1/50400)*V(4)-(1/64800)*V(2)-(1/129600)*V(1) ;...
-(13/12600)*V(3)-(1/2016)*V(4)-(1/2016)*V(2)-(19/25200)*V(1),...
-(13/75600)*V(3)-(1/12096)*V(4)-(1/9072)*V(2)-(89/907200)*V(1),...
-(1/3780)*V(3)-(1/8400)*V(4)-(1/8400)*V(2)-(1/6300)*V(1),...
-(13/75600)*V(3)-(1/9072)*V(4)-(1/12096)*V(2)-(89/907200)*V(1),...
-(19/50400)*V(3)-(23/151200)*V(4)-(31/75600)*V(2)-(23/151200)*V(1),...
(1/9450)*V(3)+(13/302400)*V(4)+(1/9450)*V(2)+(13/302400)*V(1),...
-(1/15120)*V(3)-(1/50400)*V(4)-(1/16800)*V(2)-(1/50400)*V(1),...
-(1/50400)*V(3)-(1/64800)*V(4)-(1/43200)*V(2)-(1/129600)*V(1),...
-(1/560)*V(3)-(1/1890)*V(4)-(1/1890)*V(2)-(1/1890)*V(1),...
-(1/7560)*V(3)-(1/30240)*V(4)-(1/15120)*V(2)-(1/30240)*V(1),...
(1/2520)*V(3)+(1/7560)*V(4)+(1/7560)*V(2)+(1/7560)*V(1),...
-(1/7560)*V(3)-(1/15120)*V(4)-(1/30240)*V(2)-(1/30240)*V(1),...
-(19/50400)*V(3)-(31/75600)*V(4)-(23/151200)*V(2)-(23/151200)*V(1),...
-(1/50400)*V(3)-(1/43200)*V(4)-(1/64800)*V(2)-(1/129600)*V(1),...
-(1/15120)*V(3)-(1/16800)*V(4)-(1/50400)*V(2)-(1/50400)*V(1),...
(1/9450)*V(3)+(1/9450)*V(4)+(13/302400)*V(2)+(13/302400)*V(1) ;...
(1/3360)*V(3)+(1/4725)*V(4)+(1/8400)*V(2)+(1/6048)*V(1),...
(1/20160)*V(3)+(1/28350)*V(4)+(1/37800)*V(2)+(19/907200)*V(1),...
(1/12600)*V(3)+(1/18900)*V(4)+(1/33600)*V(2)+(11/302400)*V(1),...
(1/15120)*V(3)+(1/18900)*V(4)+(1/37800)*V(2)+(1/32400)*V(1),...
(1/12600)*V(3)+(1/18900)*V(4)+(11/151200)*V(2)+(1/37800)*V(1),...
-(1/43200)*V(3)-(1/64800)*V(4)-(1/50400)*V(2)-(1/129600)*V(1),...
(1/75600)*V(3)+(1/151200)*V(4)+(1/100800)*V(2)+(1/302400)*V(1),...
(1/151200)*V(3)+(1/151200)*V(4)+(1/151200)*V(2)+(1/453600)*V(1),...
(1/1680)*V(3)+(1/3780)*V(4)+(1/7560)*V(2)+(1/7560)*V(1),...
(1/30240)*V(3)+(1/75600)*V(4)+(1/75600)*V(2)+(1/151200)*V(1),...
-(1/7560)*V(3)-(1/15120)*V(4)-(1/30240)*V(2)-(1/30240)*V(1),...
(1/15120)*V(3)+(1/25200)*V(4)+(1/75600)*V(2)+(1/75600)*V(1),...
(11/50400)*V(3)+(1/3780)*V(4)+(11/151200)*V(2)+(11/151200)*V(1),...
(1/100800)*V(3)+(1/75600)*V(4)+(1/151200)*V(2)+(1/302400)*V(1),...
```

```
(1/25200)*V(3)+(1/25200)*V(4)+(1/100800)*V(2)+(1/100800)*V(1),...
-(1/16800)*V(3)-(1/15120)*V(4)-(1/50400)*V(2)-(1/50400)*V(1) ;...
(1/560)*V(3)+(17/4200)*V(4)+(1/560)*V(2)+(17/6300)*V(1),...
(1/3360)*V(3)+(17/25200)*V(4)+(1/2520)*V(2)+(53/151200)*V(1),...
(1/2520)*V(3)+(17/25200)*V(4)+(1/3360)*V(2)+(53/151200)*V(1),...
(11/25200)*V(3)+(1/945)*V(4)+(11/25200)*V(2)+(11/18900)*V(1),...
(3/5600)*V(3)+(73/50400)*V(4)+(73/50400)*V(2)+(3/5600)*V(1),...
-(23/151200)*V(3)-(31/75600)*V(4)-(19/50400)*V(2)-(23/151200)*V(1),...
(1/18900)*V(3)+(11/151200)*V(4)+(1/12600)*V(2)+(1/37800)*V(1),...
(11/151200)*V(3)+(1/3780)*V(4)+(11/50400)*V(2)+(11/151200)*V(1),...
(73/50400)*V(3)+(73/50400)*V(4)+(3/5600)*V(2)+(3/5600)*V(1),...
(1/12600)*V(3)+(11/151200)*V(4)+(1/18900)*V(2)+(1/37800)*V(1),...
-(19/50400)*V(3)-(31/75600)*V(4)-(23/151200)*V(2)-(23/151200)*V(1),...
(11/50400)*V(3)+(1/3780)*V(4)+(11/151200)*V(2)+(11/151200)*V(1),...
(43/5040)*V(4)+(11/5040)*V(2)+(11/5040)*V(1)+(11/5040)*V(3),...
(1/7560)*V(1)+(1/1680)*V(4)+(1/3780)*V(2)+(1/7560)*V(3),...
(1/7560)*V(1)+(1/7560)*V(2)+(1/3780)*V(3)+(1/1680)*V(4),...
-(1/560)*V(4)-(1/1890)*V(2)-(1/1890)*V(1)-(1/1890)*V(3) ;...
(1/8400)*V(3)+(1/3360)*V(4)+(1/4725)*V(2)+(1/6048)*V(1),...
(1/37800)*V(3)+(1/15120)*V(4)+(1/18900)*V(2)+(1/32400)*V(1),...
(1/37800)*V(3)+(1/20160)*V(4)+(1/28350)*V(2)+(19/907200)*V(1),...
(1/33600)*V(3)+(1/12600)*V(4)+(1/18900)*V(2)+(11/302400)*V(1),...
(11/151200)*V(3)+(11/50400)*V(4)+(1/3780)*V(2)+(11/151200)*V(1),...
-(1/50400)*V(3)-(1/16800)*V(4)-(1/15120)*V(2)-(1/50400)*V(1),...
(1/151200)*V(3)+(1/100800)*V(4)+(1/75600)*V(2)+(1/302400)*V(1),...
(1/100800)*V(3)+(1/25200)*V(4)+(1/25200)*V(2)+(1/100800)*V(1),...
(11/151200)*V(3)+(1/12600)*V(4)+(1/18900)*V(2)+(1/37800)*V(1),...
(1/151200)*V(3)+(1/151200)*V(4)+(1/151200)*V(2)+(1/453600)*V(1),...
-(1/50400)*V(3)-(1/43200)*V(4)-(1/64800)*V(2)-(1/129600)*V(1),...
(1/100800)*V(3)+(1/75600)*V(4)+(1/151200)*V(2)+(1/302400)*V(1),...
(1/7560)*V(1)+(1/1680)*V(4)+(1/3780)*V(2)+(1/7560)*V(3),...
(1/75600)*V(1)+(1/15120)*V(4)+(1/75600)*V(3)+(1/25200)*V(2),...
(1/151200)*V(1)+(1/30240)*V(4)+(1/75600)*V(2)+(1/75600)*V(3),...
-(1/30240)*V(1)-(1/7560)*V(4)-(1/15120)*V(2)-(1/30240)*V(3) ;...
(1/4725)*V(3)+(1/3360)*V(4)+(1/8400)*V(2)+(1/6048)*V(1),...
(1/28350)*V(3)+(1/20160)*V(4)+(1/37800)*V(2)+(19/907200)*V(1),...
(1/18900)*V(3)+(1/15120)*V(4)+(1/37800)*V(2)+(1/32400)*V(1),...
(1/18900)*V(3)+(1/12600)*V(4)+(1/33600)*V(2)+(11/302400)*V(1),...
(1/18900)*V(3)+(1/12600)*V(4)+(11/151200)*V(2)+(1/37800)*V(1),...
-(1/64800)*V(3)-(1/43200)*V(4)-(1/50400)*V(2)-(1/129600)*V(1),...
(1/151200)*V(3)+(1/151200)*V(4)+(1/151200)*V(2)+(1/453600)*V(1),...
(1/151200)*V(3)+(1/75600)*V(4)+(1/100800)*V(2)+(1/302400)*V(1),...
(1/3780)*V(3)+(11/50400)*V(4)+(11/151200)*V(2)+(11/151200)*V(1),...
(1/75600)*V(3)+(1/100800)*V(4)+(1/151200)*V(2)+(1/302400)*V(1),...
-(1/15120)*V(3)-(1/16800)*V(4)-(1/50400)*V(2)-(1/50400)*V(1),...
(1/25200)*V(3)+(1/25200)*V(4)+(1/100800)*V(2)+(1/100800)*V(1),...
(1/7560)*V(1)+(1/7560)*V(2)+(1/3780)*V(3)+(1/1680)*V(4),...
(1/151200)*V(1)+(1/30240)*V(4)+(1/75600)*V(2)+(1/75600)*V(3),...
(1/75600)*V(1)+(1/75600)*V(2)+(1/15120)*V(4)+(1/25200)*V(3),...
-(1/30240)*V(1)-(1/30240)*V(2)-(1/15120)*V(3)-(1/7560)*V(4) ;...
-(1/2016)*V(3)-(13/12600)*V(4)-(1/2016)*V(2)-(19/25200)*V(1),...
-(1/12096)*V(3)-(13/75600)*V(4)-(1/9072)*V(2)-(89/907200)*V(1),...
-(1/9072)*V(3)-(13/75600)*V(4)-(1/12096)*V(2)-(89/907200)*V(1),...
-(1/8400)*V(3)-(1/3780)*V(4)-(1/8400)*V(2)-(1/6300)*V(1),...
-(23/151200)*V(3)-(19/50400)*V(4)-(31/75600)*V(2)-(23/151200)*V(1),...
(13/302400)*V(3)+(1/9450)*V(4)+(1/9450)*V(2)+(13/302400)*V(1),...
-(1/64800)*V(3)-(1/50400)*V(4)-(1/43200)*V(2)-(1/129600)*V(1),...
-(1/50400)*V(3)-(1/15120)*V(4)-(1/16800)*V(2)-(1/50400)*V(1),...
-(31/75600)*V(3)-(19/50400)*V(4)-(23/151200)*V(2)-(23/151200)*V(1),...
```

```
        -(1/43200)*V(3)-(1/50400)*V(4)-(1/64800)*V(2)-(1/129600)*V(1),...
        (1/9450)*V(3)+(1/9450)*V(4)+(13/302400)*V(2)+(13/302400)*V(1),...
        -(1/16800)*V(3)-(1/15120)*V(4)-(1/50400)*V(2)-(1/50400)*V(1),...
        -(1/560)*V(4)-(1/1890)*V(2)-(1/1890)*V(1)-(1/1890)*V(3),...
        -(1/30240)*V(1)-(1/7560)*V(4)-(1/15120)*V(2)-(1/30240)*V(3),...
        -(1/30240)*V(1)-(1/30240)*V(2)-(1/15120)*V(3)-(1/7560)*V(4),...
        (1/2520)*V(4)+(1/7560)*V(2)+(1/7560)*V(1)+(1/7560)*V(3)];
    translate = zeros(16,16);
    translate(1,1) = 1;
    translate(2:4,2:4) = J;
    translate(5,5) = 1;
    translate(6:8,6:8) = J;
    translate(9,9) = 1;
    translate(10:12,10:12) = J;
    translate(13,13) = 1;
    translate(14:16,14:16) = J;
    elem_V = det(J)*(translate'*elem_V*translate);
end
```

B.4.4 Elem_matrix_E_3D

```
function [ elem_E ] = Elem_matrix_E_3D( J,ndof )
%Elem_matrix_E_3D Computes the elemental overlap matrix
%   The elemental overlap energy matrix is formed using either linear
%   basis functions or cubic basis functions.
%
%   [ELEM_E]=ELEM_MATRIX_E_3D(J,NDOF)
%      ELEM_E: ELEMENTAL OVERLAP ENERGY MATRIX (4 X 4 OR 16 X 16)
%      J: JACOBIAN MATRIX (3 X 3)
%      NDOF: DEGREE OF FREEDOM PER NODE
%-------------------------------------------------------------------------
%
%   Variables:
%       Translate: Matrix that converts ELEM_E from local coordinates to
%       global coordinates.  This is only used when ndof does not equal 1.
%
%Linear shape functions are defined as;
%   N_1(x,y)=1-x-y-z;     0<=x,y,z<=1
%   N_2(x,y)=x;           0<=x,y,z<=1
%   N_3(x,y)=y;           0<=x,y,z<=1
%   N_4(x,y)=z;           0<=x,y,z<=1
%
%Cubic shape functions are defined as;
%   N_1(x,y,z)=1-N_5-N_9-N_13;                      0<=x,y,z<=1
%   N_2(x,y,z)=x-N_5-N_6-N_10-N_14;                 0<=x,y,z<=1
%   N_3(x,y,z)=y-N_7-N_9-N_11-N_15;                 0<=x,y,z<=1
%   N_4(x,y,z)=z-N_8-N_12-N_13-N_16                 0<=x,y,z<=1
%   N_5(x,y,z)=3*x^2-2*x^3;                         0<=x,y,z<=1
%   N_6(x,y,z)=-x^2+x^3;                            0<=x,y,z<=1
%   N_7(x,y,z)=x^2*y;                               0<=x,y,z<=1
%   N_8(x,y,z)=x^2*z;                               0<=x,y,z<=1
%   N_9(x,y,z)=3*y^2-2*y^3;                         0<=x,y,z<=1
%   N_10(x,y,z)=x*y^2;                              0<=x,y,z<=1
%   N_11(x,y,z)=-y^2+y^3;                           0<=x,y,z<=1
%   N_12(x,y,z)=y^2*z;                              0<=x,y,z<=1
%   N_13(x,y,z)=3*z^2-2*z^3;                        0<=x,y,z<=1
%   N_14(x,y,z)=x*z^2;                              0<=x,y,z<=1
%   N_15(x,y,z)=y*z^2;                              0<=x,y,z<=1
%   N_16(x,y,z)=-z^2+z^3;                           0<=x,y,z<=1
```

```
%-----------------------------------------------------------------------
%
if (ndof == 1)
    elem_E = 1/120*[2 1 1 1; 1 2 1 1; 1 1 2 1; 1 1 1 2];
    elem_E = det(J)*elem_E;
else
    elem_E = [1/28, 13/2520, 13/2520, 13/2520, 13/1260, -1/360, 1/1260,...
              1/1260, 13/1260, 1/1260, -1/360, 1/1260, 13/1260, 1/1260,...
              1/1260, -1/360 ; 13/2520, 11/11340, 1/1260, 1/1260,...
              19/7560, -1/1512, 1/5040, 1/5040, 13/7560, 1/5670, -1/2160,...
              1/7560, 13/7560, 1/5670, 1/7560, -1/2160 ; 13/2520, 1/1260,...
              11/11340, 1/1260, 13/7560, -1/2160, 1/5670, 1/7560, 19/7560,...
              1/5040, -1/1512, 1/5040, 13/7560, 1/7560, 1/5670, -1/2160 ;...
              13/2520, 1/1260, 1/1260, 11/11340, 13/7560, -1/2160,...
              1/7560, 1/5670, 13/7560, 1/7560, -1/2160, 1/5670, 19/7560,...
              1/5040, 1/5040, -1/1512 ; 13/1260, 19/7560, 13/7560,...
              13/7560, 19/1260, -17/5040, 17/15120, 17/15120, 1/252,...
              19/30240, -11/10080, 1/4320, 1/252, 19/30240, 1/4320,...
              -11/10080 ; -1/360, -1/1512, -1/2160, -1/2160, -17/5040,...
              1/1260, -1/3780, -1/3780, -11/10080, -1/6048, 1/3360,...
              -1/15120, -11/10080, -1/6048, -1/15120, 1/3360 ; 1/1260,...
              1/5040, 1/5670, 1/7560, 17/15120, -1/3780, 1/7560,...
              1/15120, 19/30240, 1/10080, -1/6048, 1/30240, 1/4320,...
              1/30240, 1/45360, -1/15120 ; 1/1260, 1/5040, 1/7560,...
              1/5670, 17/15120, -1/3780, 1/15120, 1/7560, 1/4320,...
              1/30240, -1/15120, 1/45360, 19/30240, 1/10080, 1/30240,...
              -1/6048 ; 13/1260, 13/7560, 19/7560, 13/7560, 1/252,...
              -11/10080, 19/30240, 1/4320, 19/1260, 17/15120, -17/5040,...
              17/15120, 1/252, 1/4320, 19/30240, -11/10080 ; 1/1260,...
              1/5670, 1/5040, 1/7560, 19/30240, -1/6048, 1/10080,...
              1/30240, 17/15120, 1/7560, -1/3780, 1/15120, 1/4320,...
              1/45360, 1/30240, -1/15120 ; -1/360, -1/2160, -1/1512,...
              -1/2160, -11/10080, 1/3360, -1/6048, -1/15120, -17/5040,...
              -1/3780, 1/1260, -1/3780, -11/10080, -1/15120, -1/6048,...
              1/3360 ; 1/1260, 1/7560, 1/5040, 1/5670, 1/4320, -1/15120,...
              1/30240, 1/45360, 17/15120, 1/15120, -1/3780, 1/7560,...
              19/30240, 1/30240, 1/10080, -1/6048 ; 13/1260, 13/7560,...
              13/7560, 19/7560, 1/252, -11/10080, 1/4320, 19/30240,...
              1/252, 1/4320, -11/10080, 19/30240, 19/1260, 17/15120,...
              17/15120, -17/5040 ; 1/1260, 1/5670, 1/7560, 1/5040,...
              19/30240, -1/6048, 1/30240, 1/10080, 1/4320, 1/45360,...
              -1/15120, 1/30240, 17/15120, 1/7560, 1/15120, -1/3780 ;...
              1/1260, 1/7560, 1/5670, 1/5040, 1/4320, -1/15120,...
              1/45360, 1/30240, 19/30240, 1/30240, -1/6048, 1/10080,...
              17/15120, 1/15120, 1/7560, -1/3780 ; -1/360, -1/2160,...
              -1/2160, -1/1512, -11/10080, 1/3360, -1/15120, -1/6048,...
              -11/10080, -1/15120, 1/3360, -1/6048, -17/5040, -1/3780,...
              -1/3780, 1/1260];
    translate = zeros(16,16);
    translate(1,1) = 1;
    translate(2:4,2:4) = J;
    translate(5,5) = 1;
    translate(6:8,6:8) = J;
    translate(9,9) = 1;
    translate(10:12,10:12) = J;
    translate(13,13) = 1;
    translate(14:16,14:16) = J;
    elem_E = det(J)*(translate'*elem_E*translate);
end
```

B.4.5 Apply_interface_bc_3D

```
function [ elem_A ] = apply_interface_bc_3D( elem_A,int_nodes,nd,m )
%Apply_interface_bc Applies the boundary conditions on the interface
%between the two materials using the method as outlined my Ram-Mohan.
%
%   [GLOB_A]=APPLY_BC(GLOB_A,INT_NODES,ND,M)
%      Elem_A: ELEMENTAL MATRIX
%      INT_NODES: VECTOR CONTAINING NODES ON THE INTERFACE BETWEEN THE TWO
%      MATERIALS
%      ND: LOCAL OR ELEMENTAL NODE INDEXING (1 X 4)
%      M: EFFECTIVE MASS
%-------------------------------------------------------------------------
%
ind = find(ismember(nd,int_nodes));
if (~isempty(ind))
    k=0;
    for i=1:length(ind)
    start=(ind(i)-1)*4;
        for j=2:4;
            k=k+1;
            e_index(k)=start+j;
        end
    end
    elem_A(:,e_index) = 1/m*elem_A(:,e_index);
    elem_A(e_index,:) = 1/m*elem_A(e_index,:);
end
end
```

B.4.6 Index_3D

```
function [ index ] = index_3D( nd,ndof )
%Index_3D Assigns the system degree of freedom to the element node
%   Index is system dof vector which can be used to place the elements
%   associated with element matrices in the global matrices.
%
%   [INDEX]=INDEX_3D(ND,NNEL,NDOF)
%      INDEX: SYSTEM DOF VECTOR
%      ND: VECTOR CONTAINING GLOBAL NODE NUMBERS(1 X 4)
%      NDOF: DEGREE OF FREEDOM PER NODE
%-------------------------------------------------------------------------
%
start=(nd-1)*ndof+1;
if (ndof == 1)
    index = start;
else
    index = [start ; start+1 ; start+2 ; start+3];
    index = index(:)';
end
```

B.5 Post Processing Code

```
function [ E psi ] = Energy_psi( glob_K,glob_V,glob_E,ndof,V )
%Energy_psi Computes the energy eigenvalues and wavefuntion
%   Using the global matricies obtained from processing code, this function
%   will compute the energy and normalized wavefunction.  The energy levels
%   are sorted from lowest to highest.
%
%   [E PSI]=ENERGY_PSI ( GLOB_K,GLOB_V,GLOB_E,NDOF,V )
```

```
%      E: ENERGY EIGENVALUES
%      PSI: NORMALIZED WAVEFUNCTION
%      GLOB_K: GLOBAL KINETIC ENERGY MATRIX
%      GLOB_V: GLOBAL POTENTIAL ENERGY MATRIX
%      GLOB_E: GLOBAL OVERLAP MATRIX
%      NDOF:  DEGREES OF FREEDOM PER NODE
%      V: VALUE OF SMALLEST POTENTIAL VALUE IN MESH
%-------------------------------------------------------------------------
%   Functions:
%      SPTARN: Eigenvalues and eigenvectors of sparse matrix in interval
%      [0.1*V,V].

[psi E result] = sptarn(glob_K+glob_V,glob_E,0.1*V,V);
psi = psi(1:ndof:end,:);
for i=1:length(psi(1,:))
    normal = norm(psi(:,i));
    psi(:,i) = psi(:,i)./normal;
end
end
```

B.5.1 Simp_psi_plot

```
function simp_psi_plot( gcoord,psi,n )
%simp_psi_plot Plots the 2D wavefuntion corresponding to the energy
%eigvenvalue number obtained from Energy_psi.
%
%   SIMP_PSI_PLOT ( GCOORD,PSI,N )
%      GCOORD: COORDINATES OF MESH
%      PSI: WAVEFUNCTION
%      N: ENERGY LEVEL NUMBER (OBTAINED FROM ENERGY_PSI)
%-------------------------------------------------------------------------
%   Variables:
%       F: interpolated function defined by coordinates of mesh corresponding
%       to the values of the wavefunction.
%       i: increment in x direction
%       j: increment in y direction
%       x: x coordinates used in the surface plot
%       y: y coordinates used in the surface plot
%       Z: interpolated wavefunction values at x and y coordinates
%   Functions:
%       TriScatteredInterp: Interpolate scattered data
%       Meshgrid: Generate X and Y  arrays for 3D plots
%       Surf: 3D shaded surface plot
F = TriScatteredInterp(gcoord(:,1),gcoord(:,2),psi(:,n));
i = (max(gcoord(:,1))-min(gcoord(:,1)))/50;
j = (max(gcoord(:,2))-min(gcoord(:,2)))/50;
x = min(gcoord(:,1)):i:max(gcoord(:,1));
y = min(gcoord(:,2)):j:max(gcoord(:,2));
[X Y] = meshgrid(x,y);
Z = F(X,Y);
surf(X,Y,Z);
end
```

Vita

Steven Evans Jenks was born in Washington Twp., New Jersey on January 31, 1980. He attended The College of New Jersey from 1998 - 2002 and received a Bachelor of Science in Engineering Science. He began his graduate studies at Drexel University in September 2004 and received a Masters of Science in physics in June 2007. Under the direction of Dr. Robert Gilmore, he will complete his Doctor of Philosophy in physics in 2012.

A notable list of his accomplishments and acknowledgments during his time at Drexel University is as follows:

- *Quantum dot solar cell: Materials that produce two intermediate bands*, with Robert Gilmore, J. Renewable Sustainable Energy, **2**, 013111 (2010).
- *Selecting semiconductor materials for the quantum dot solar cell*, with Robert Gilmore, J. Renewable Sustainable Energy (to be submitted).
- Asked to speak about our research at the 2011 and 2012 CMOS emerging technologies conference (see www.cmoset.com).

  > Dear Dr. Jenks, I am chairing emerging technologies meeting in Whistler 2011, attached or www.cmoset.com. I have read with great interests your QD paper. Would you be interested in giving a talk at our event?...best regards, Kris
  >
  > Steve, we are starting to look for 2012 speakers for Vancouver meeting, attached, perhaps you can suggest some people to contact or might be interested in giving a talk yourself...best regards, Kris

- Asked to contribute a chapter in a Microsystems textbook.

  > Any interests in contributing a chapter to the Microsystems book I am editing?...best regards, Kris

- I have reviewed a manuscript in the journal IEEE Electron Device Letters.

  > Dear Dr. Jenks:
  >
  > Based on your expertise and specialized knowledge, I would like to invite you to review the above mentioned manuscript (see the abstract below) submitted to the IEEE Electron Device Letters. As EDL is a fast-turnaround journal, it is very important that we receive your comments within one (1) week.
  >
  > Please respond to my invitation as soon as possible by clicking the appropriate link below to automatically record your reply within our online system, ScholarOne Manuscripts. If you are unable to review at this time, your recommendation of another expert reviewer would be appreciated.
  >
  > Once you accept, you will be notified by e-mail about how to access the paper.
  >
  > Thank you for your contribution in maintaining the high standards of the Journal.
  >
  > Sincerely, Prof. Jesus del Alamo EDL Editor alamo@mit.edu