WIDE BAND GAP ENGINEERING OF MAGNESIUM OXIDE-ZINC OXIDE II-VI SEMICONDUCTORS

By Robert Matthew Leone

A Thesis

Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Science

in Applied Physics

Northern Arizona University

May 2006

Abstract

Wide Band Gap Engineering of

Magnesium Oxide-Zinc Oxide II-VI Semiconductors

Robert Matthew Leone

Wide-gap semiconducting materials are extending critical applications in electronics and optoelectronics with the continued advancement of blue to ultraviolet LEDs and lasers. Magnesium oxide-zinc oxide (MgO-ZnO) is a II-VI semiconducting material with a wide band gap. Its alloys have been increasingly investigated due to a UV luminescence ranging from 150 to 400 nm or alternatively wide band gaps from 3.3 to 7.8 eV. Additionally, advances in thin-film deposition technology increasingly allow for directed layered construction of semiconductor materials. This research was directed at finding an accurate model to predict band gaps for B1-type lattices of the MgO-ZnO alloys in order to direct experimental construction of this material to achieve specific band gaps.

A first-principles model band gap functional has been developed herein that predicts the band gaps of cubic MgO-ZnO alloys for cell shape, size and atomic configuration. Specifically, LDA first-principles Γ-point band gap energies were used as input to construct an Ising-like cluster expansion of the B1-type alloys. This construction utilized a novel genetic algorithm to determine which cluster types to be used in the expansion.

Once obtained, the cluster expansion was used to predict the band gap energies of all cubic unit cells with 13 cation sites or less over 16,000 unique structures. The results of these predictions were analyzed to determine configurations that resulted in specific band gaps, particularly reasonably simple structures for use in band gap engineering. This model therefore allows for the *design* of specific wide band gap MgO-ZnO alloy superlattices to achieve targeted band gaps.

Specific predictions include a band gap of 7.5 eV for the MgO_5-ZnO_1 superlattice in the (320) direction, a 6.8 eV band gap for the MgO_5-ZnO_1 superlattice in the (211) direction, and a band gap of 6.07 eV MgO_3-ZnO_2 superlattice in the (100) direction. Results also show that for MgO_x-ZnO_1 superlattice stacking in the (100) direction, the band gap energy approaches a limit with increasing x well below the band gap of pure MgO indicating charge localization on the constituent Zn.

Acknowledgements

I would like to thank my advisor, Gus Hart, for the opportunity to conduct intensive solid state research, to travel and present talks at conferences, to publish, to learn a great deal of computational physics, and to act as a mentor in developing and planning my career as a physicist. I would also like to thank the entire staff and faculty of the Physics and Astronomy Department for their generous support and encouragement. Finally, I would like to thank my wife, Sarah, for helping me to realize my dream to become a practicing physicist.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Semiconductor Physics

Solid state physics provides a theoretical framework for understanding the properties of semiconductors via band theory. This theory predicts that energy eigenvalues of the solutions for the electronic structure governed by the Schroedinger wave equation (SWE) will be grouped together into nearly continuous distributions called bands.

The energy bands are a result of the calculation of energy eigenstates for a periodic potential with the periodic boundary condition that the resultant wave function solution must obtain the same value except for a phase shift upon translation of the wave function by an integer number of lattice vectors. Restated, the probability density of the wave function must be equal at equivalent points of the periodic lattice. One may then view the resultant wave function as being composed of a free electron wave function modulated by a wave function that satisfies the periodic boundary conditions. This is called a Bloch function and has the form,

$$\psi_{\bar{k}}(\bar{r}) = e^{i\bar{k}\bullet\bar{r}} u_{\bar{k}}(\bar{r}),$$

where

$$u_{\bar{k}}(\bar{r}) = u_{\bar{k}}(\bar{r} + \bar{T})$$

for any lattice translation, **T**.[1]

The energies between bands will not be eigenvalues of the system's SWE and therefore there will be no corresponding eigenstates to yield these energies. The highest energy band with occupied eigenstates is the valence band of the system while the lowest energy unoccupied band is the conduction band. The difference in energy between the highest energy eigenvalue of the valence band and the lowest energy eigenvalue of the conduction band is termed the band gap energy. This purely quantum mechanical effect explains the difference between metals and semiconductors, where the former has a partially filled valence band and the latter a completely filled valence band with an energy gap between the highest energy eigenstate of the valence band and the lowest unoccupied energy eigenstate of the conduction band.

Energy bands in solids are typically depicted as dispersion graphs, i.e. in terms of energy versus the wave vector energy quantum number, **k**, due to the unique identification of eigenstates according to the band quantum number and energy quantum number (as in Figure 1). The band plot may be considered a dispersion graph, a graph of energy versus momentum, because each wave vector energy quantum number, **k**, that uniquely defines an energy eigenstate is also an eigenstate of the momentum operator. A simple relationship describes the momentum eigenvalue in a simple way,

$$\bar{p}_k = \frac{h}{2\pi}\bar{k}\,.$$

One must be careful to point out that this is truly the *crystal momentum*, i.e. the momentum operator acting on only the Bloch function of the total wavefunction, and is only equal to the true electron momentum for the case of free electrons.

Figure 1 shows the band structure of MgO calculated using density functional theory (DFT) within the local density approximation (LDA). Only high symmetry points of the Brillouin zone are labeled.

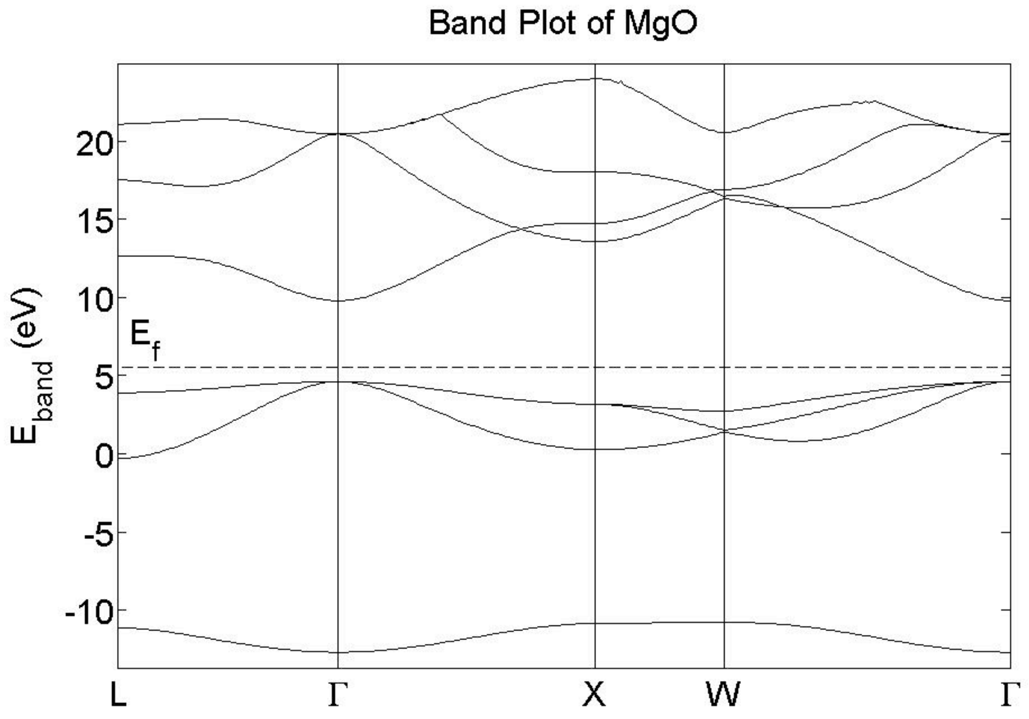


Figure 1: Band structure of MgO in the rocksalt crystal structure.

These calculations are made using pseudopotentials wherein the effects of core electrons that are assumed to remain unchanged during chemical bonding are combined with the electrostatic potentials of the nuclei to form a combined potential. Therefore, only outer valence electrons from each atom are treated quantum mechanically, i.e. wave functions calculated from the modified SWE using pseudopotentials and the LDA.

In the case of MgO, the magnesium atom is calculated using only its valence $3s^2$ electrons and the $2s^2 3p^4$ valence electrons for oxygen are used. Thus in terms of tight binding theory, these four atomic states may hybridize to form four new states in each primitive cell, and so four fully occupied bands are seen below the Fermi energy in the band plot of Figure 1.

In a semiconductor at 0 K, the valence band is completely occupied and the conduction band is empty. In a direct band gap material, the energy difference between these bands is smallest at the Γ- point, i.e. **k**=0 in reciprocal space, and it is this difference that is called the band gap. Electrons may be excited from the valence band to the conduction band only if enough energy can be provided through electron-phonon interactions or by optical stimulation. Because the band gap is direct and because the photon momentum changes the electron momentum very little, optical transitions will occur nearly vertically on in the band plot. Only through interaction with the crystal lattice momentum can the electron's momentum be substantially changed so that transitions from one eigenstate of the valence band with one particular quantum number **k** to a different quantum number in the conduction band occur, as would be necessary for *indirect* band gap materials.

A small number of the semiconductor electrons may be excited to the lowest energy states of the conduction band. These conduction electrons have a vast number of empty eigenstates of nearly equal energy through which to move in the conduction band due to the large density of states of the conduction band at the Γ-point. The conduction band electrons leave behind empty eigenstates in the valence band described as holes. These empty valence eigenstates are surrounded by a great number of occupied valence eigenstates of nearly equal energy due to the large density of states of the valence band at the Γ-point. Upon reorganization of the remaining valence electrons in these nearby eigenstates, the holes may be shifted around the valence band from one eigenstate to another. These two simultaneous effects explain the existence of charge carriers in undoped semiconductors.

An energy equal or greater than that of the band gap must be provided to an electron in order for it to be excited to the conduction band. At room temperature, the thermal energy per electron is about 1/40 eV, not nearly enough to span the typical semiconductor band gap of 1-5 eV. Statistically, however, there are a great number of potential charge carriers. According to Maxwell-Boltzmann statistics which apply to fermions only at very high temperatures, the probability of a thermal electron obtaining a 1 eV increase in energy is given as $e^{-40} = 10^{-17.4}$. For 1 mole of valence electrons in eigenvalue energies near the top of the valence band, there will be $10^{-17.4} * 10^{24.8} = 10^{7.4}$ conduction electrons with enough energy to be excited so that statistical mechanics predicts available charge carriers in a semiconductor at room temperature. Fermi-Dirac statistics gives a larger value of approximately 10^{10} for the number of charge carriers excited across a 1 eV band gap per mole.[2]

1.2 MgO-ZnO Alloying

Wide band gap materials are important for creating UV diodes and UV lasers that can greatly increase data storage density, kill bacteria, and detect high temperature burning among a myriad of other useful semiconducting devices. Alloys made from MgO and ZnO give wide band gap semiconducting materials with highly tunable band gaps.

ZnO favors a wurtzite crystal structure while MgO forms in the rocksalt structure. However, the formation energies of structures of MgO-ZnO alloys in the wurtzite form and rocksalt form have been found to be very similar suggesting growth of alloys of either structure is possible.[3]

Current laboratory growth techniques to produce MgO-ZnO alloys have used pulsed laser deposition with a sapphire substrate (Al_2O_3) directed along the (0001) plane,[4,5,6,7] on a MgO (100) substrate,[5] on Si (100) with a $SrTiO_3$ buffer,[8] on Si (100) with a TiN buffer,[5] using radio frequency two-source magnetron sputtering on a fused silica substrate,[9] and using direct current magnetron sputtering on quartz and soda-lime glass substrates.[10] Experimental analysis of $MgO_{1-x}ZnO_x$ alloys show wurtzite formation for large ZnO concentration,[6,7,9] and rocksalt formation with high MgO concentration,[5,6,8,10] and an intermediate range of ZnO concentrations for which phase separation occurs. Rocksalt alloying has been reported as high as 50% ZnO [5] while wurtzite for greater than 54% ZnO .[9] Each of these reported alloying experiments produced band gap estimates as a function of ZnO concentration that were used to adjust the theoretical calculations performed in this research.

Chapter 2

The computational procedure

2.1 First Principles Calculations

Electronic structure calculations were performed using the Vienna Ab-initio Software Package (VASP). VASP computes the electronic structure of a solid using Density Functional Theory (DFT) to minimize the Gibbs free energy by relaxing the atomic positions and solving the Kohn-Sham equations to find the electronic structure. That is it solves the quantum system of positive ionic potentials surrounded by negative electrons by calculating free energies in terms of the single variable of the density of electrons as a function of position.[11,12] This is done rather than solving the many-body Schrödinger equation (impossible) or by constructing an approximate many-body wavefunction using a Slater determinant.

VASP uses the Local Density Approximation wherein the exchange and correlation energy of the electrons is approximated as a local functional of the electron density, [13]

$$V_{\mathrm{xc}}(r) = \varepsilon_{\mathrm{xc}}\big(n(r)\big) + n(r)\frac{\partial \varepsilon_{\mathrm{xc}}\big(n(r)\big)}{\partial n(r)}.$$

Another approximation made in the first principles calculations is that most of the electron density will always be localized near the positive ionic potential, i.e. the core electrons will not participate in bonding. Computational speed is then increased greatly by using the ultra-soft Vanderbilt pseudopotential scheme[14] where a non-localized potential interacts with valence electrons whose wave functions are modeled with plane waves and a localized potential interacts with the core electrons described simple by the local electron density.

Finally, VASP allows relaxation of the lattice vectors until free energy minimization has been achieved. As output, one obtains among many things, the final Gibbs free energy and the Γ-point band gap energy for the system.

In this research, 105 unique MgO-ZnO alloys (different primitive cells) were calculated for the MgO-ZnO pseudobinary system in concentrations ranging from 0-100% ZnO in the rocksalt crystal structure. The structures chosen included all primitive cells comprised of 5 cations or less, 9 structures with 6 cations, 7 structures with 7 cations, and 17 structures with 8 cations. For any structure not consisting of a 50-50 ratio of Mg to Zn, two computations were performed, one for the Mg rich alloy and another for the Zn rich. Therefore, first principles calculations were made for many structures consisting of 8 cations or less. This computationally expensive data set was generated in order to obtain an accurate input database with which to construct the cluster expansion model.

2.2 Cluster Expansion

The cluster expansion model is an Ising-like model that produces a simple and fast band gap functional of structure to simulate binary alloy systems. The band gap functional of the cluster expansion takes on the form

$$E_{gap} = p_o + p_1 \sum_i S_i + \sum_{i \neq j} p_{ij} S_i S_j + \sum_{i \neq j \neq k} p_{ijk} S_i S_j S_k + \dots ,$$

where E_{gap} is the Γ-point band gap energy of the structure, S_i are the "spin" values at each lattice site, and the p's are band gap energy fitting parameters obtained from the first-principles calculations.[15] For a lattice site occupied by a type A atom, a spin value of +1 is assigned and for a type B atom, a –1 spin value is assigned. Since this model only requires the product of spin values multiplied by a constant term, predicting band gap energies with the functional is fast with the band gaps for thousands of unique structures being computed in seconds.

The first summation of the cluster expansion model functional is simply a single concentration-dependent band gap energy term. The second term describes the band gap energy associated with various pair interactions. There are many types of pair interactions that may be ordered for convenience as first nearest neighbor, second nearest neighbor, etc. As an example of how summation proceeds, for any lattice sites i and j that are first nearest neighbors, p_{ij} is the constant nearest neighbor pair band gap energy parameter for all first nearest neighbors. The summation runs over the entire lattice with only the spin values alternating between 1 and -1 so that a total first nearest neighbor band gap energy is calculated. However, this summation is not only for first nearest neighbor pairs, and there will be a different p_{ij} for every kind of pair interaction so that every type of pair interaction delivers its own total interaction band gap energy.

The third term of the model functional then represents energies from all possible three-body interactions, and so it goes for a large number of combinations for any finite cell of realistic dimensions. Clearly the astronomical number of {p} interaction parameters cannot be fit from a small number of first principles structures. Therefore, truncation of the model functional must be made so that only a handful of pair and multi-body interaction types are incorporated into the model.

Typically pair interactions up to 20-25 nearest neighbors are used in the model and for the model of the MgO-ZnO alloys, the first 24 nearest neighbors were selected. Then the selection of which kinds of multi-body ineractions must be made. Should first nearest neighbor tetrahedrons be use or a linear 3-body chain? A direct search for the best choice is combinatorially intractable, and direct numeration schemes have been shown to produce inaccurate models.[16] Once the cluster types have been selected, multivariate regression is performed on the set of first principles results yielding the interaction energies, {p}, of the cluster expansion model.

Some care must be made to limit the number of interaction types used as there are only a limited number of first principles input data structures and overfitting of the data could easily occur. Overfitting of the input data would render a cluster expansion model that would predict the input structures' band gaps perfectly, but would be useless to predict the band gaps for novel atomic arrangements.

Ninety nine first principles calculations of band gap energies were used as input for the cluster expansion model, and so the number of interaction types was to be used was limited to 45 to prevent overfitting (slightly less than 50% of the input data structures). Of these 45, one interaction represents a constant term, another a single lattice site term, then up to the first 25 nearest neighbor pairs were used, and the remaining 18 interactions are from various multi-body terms. These 18 multi-body cluster types were to be chosen from 124 possible different interaction types, thus $\sim 10^{21}$ possible combinations.

Once the multi-body terms to be used were selected, the cluster expansion model was created from the input data. This was done by using 99 first-principles calculations giving band gap energies as a function of structure, but withholding 40 groups of 5 input band gap energies. The resultant model (built from 94 input structures) was used to predict the band gaps for the 5 left-out structures in each group. The error in predicting the band gaps of the groups of 5 left-out structures then formed a measure of the predictivity of the cluster expansion model. This leave-many out cross validation has been used successfully in past research.[17,18,19] After being done for several combinations of multi-body types, the most predictive cluster expansion model was obtained.

A novel technique using a genetic algorithm has been developed to address the process of multi-body selection.[20,21] This technique searches for the set of multi-body interaction types that generate the most predictive cluster expansion model relying on the highly correlated nature of the problem.

The genetic algorithm begins with a pool of 50 individual sets of randomly selected multi-body clusters was generated and models created for each combination using the process described above. The predictivity for each of the 50 unique models was calculated. The 20 least predictive members of this set were thrown out, and 20 new members were generated to replace them by *mating* the remaining 30 members. Specifically, cluster types that were selected in highly predictive models had a higher probability of being selected again than those cluster types that had not been found in highly predictive models. In order to prevent becoming isolated in local minima, no individual selection of cluster types may be used in more than 50 iterations. For the MgO-ZnO alloy cluster expansion, 10 candidates of different multi-body selections were generated and the most predictive cluster expansion model obtained. The final model included 24 pair interactions, seven 3-body, four 4-body, two 5-body, and five 6-body interaction types.

Once the multi-bodies to be used were selected, the p band gap energy parameters were fit to all 99 first principles calculations and the model was complete. The band gaps for thousands of unique structures were calculated in seconds and their band gap energies were investigated as a function of cation arrangement.

Before this cluster expansion model was made, however, the first principles data needed to be adjusted. Band gap energies are always underestimated using LDA. The use of a local density functional causes an electron to artificially interact with itself so that there is an artificial increase in the valence band energy. To accommodate for this underestimation, a simple concentration weighted shift was applied to all calculated band gap energies so that the band gaps for the concentration endpoints of pure MgO and pure ZnO were made to match experimental measurements. A similar procedure has been conducted in the past with success.[22] This adjusted data shown in Table 1 was used to fit the cluster expansion with the genetic algorithm technique so that the predicted band gap energies corresponded to experimentally observed values.

Chapter 3

Input: The First Principles Band Gaps

Figure 1 shows the calculated band gaps MgO-ZnO from first principles for the 105 unique structures at varying concentration with the addition of the LDA calculation of pure ZnO in the wurtzite structure. Figure 2 shows how these calculations compare to experimentally measured band gaps at various concentrations. These values are shown in comparison to values that have been experimentally reported.[5,6,7,8,10,11] At 100% pure MgO, the calculated values are nearly 2.5 eV below the experimental. Pure ZnO does not form in the rocksalt structure, but rather a wurtzite structure. Here a calculation was made for ZnO in its native configuration to determine the LDA deviation from experimental values, about 2 eV too low. Additional experimental values are shown for reference, but only the endpoint experimental values were used to adjust LDA calculations since these are well established.

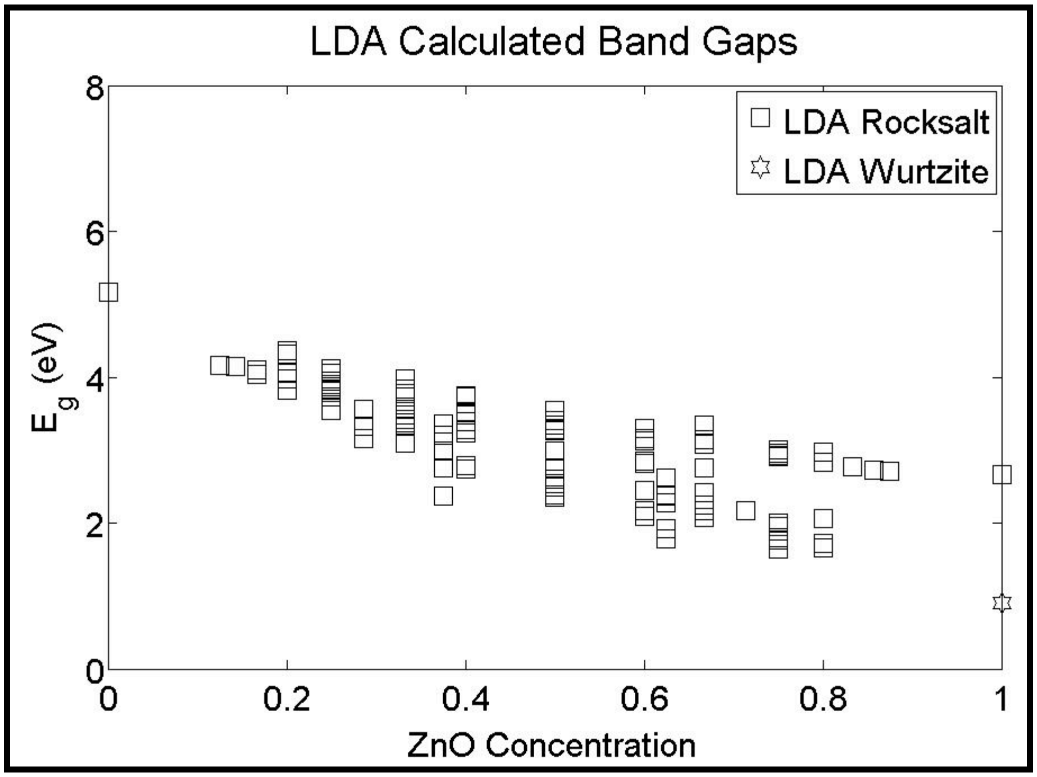


Figure 1: Unadjusted first principles band gap energies.

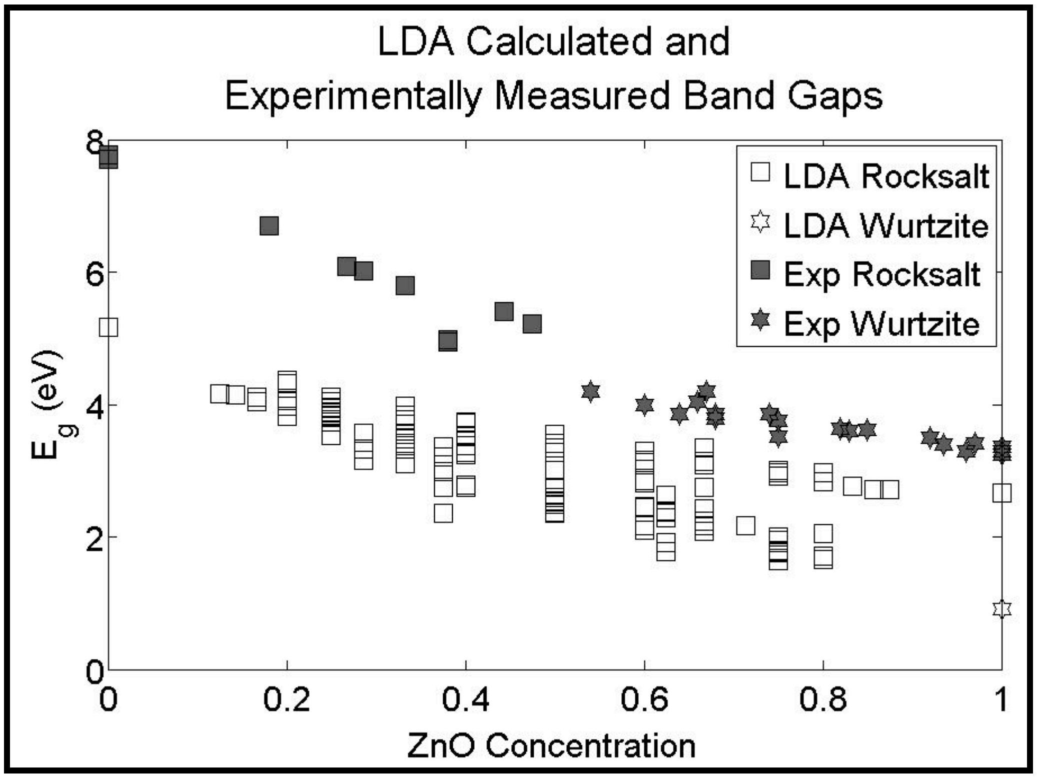


Figure 2: LDA calculated band gaps versus experimental measurements.

The deviations of the calculations at the two endpoints were then used to create a concentration weighted shift of the LDA band gap data upward so that the LDA band gaps of pure MgO and ZnO matched experiment as shown in Figure 3.

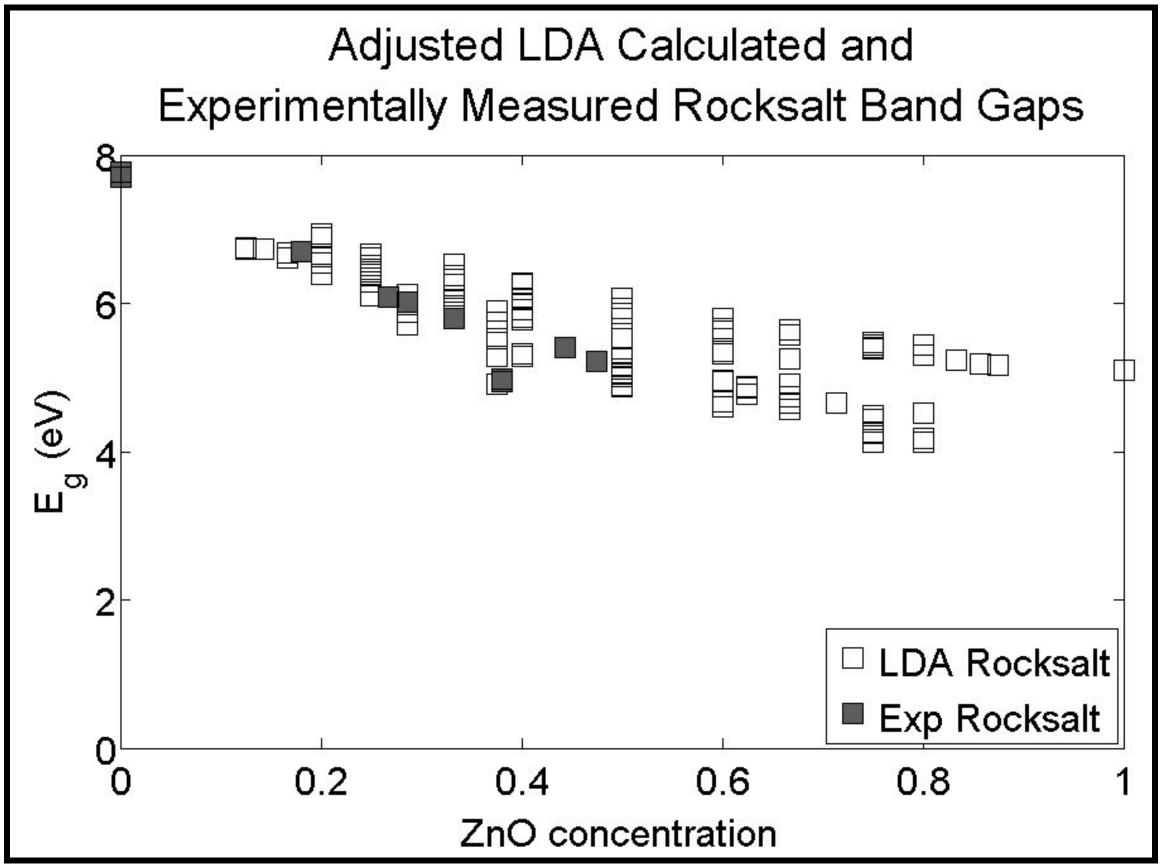


Figure 3: Adjusted first principles band gap energies.

The rocksalt experimentally measured band gaps align well with the adjusted LDA band gap data over the ZnO concentration range (0-50%) where rocksalt structure of the alloy has been synthesized.

Chapter 4

Cluster Expansion Results

4.1 Model Accuracy

Once the cluster expansion was created, it was used to predict the LDA band gaps for all the input structures that had been used to create it. What is most important here is that the predictions remain very close to the input data while having used the minimum necessary number of cluster types that capture the physics of the system.

Figure 4 shows that the performance of the cluster expansion model compared to the first principles data is reasonably close for band gap engineering. However, whereas the first principles calculations for these $MgO_{1-x}ZnO_x$ alloy structures were computationally expensive requiring thousands of hours of computation, the cluster expansion model was able to yield these predicted band gaps in seconds.

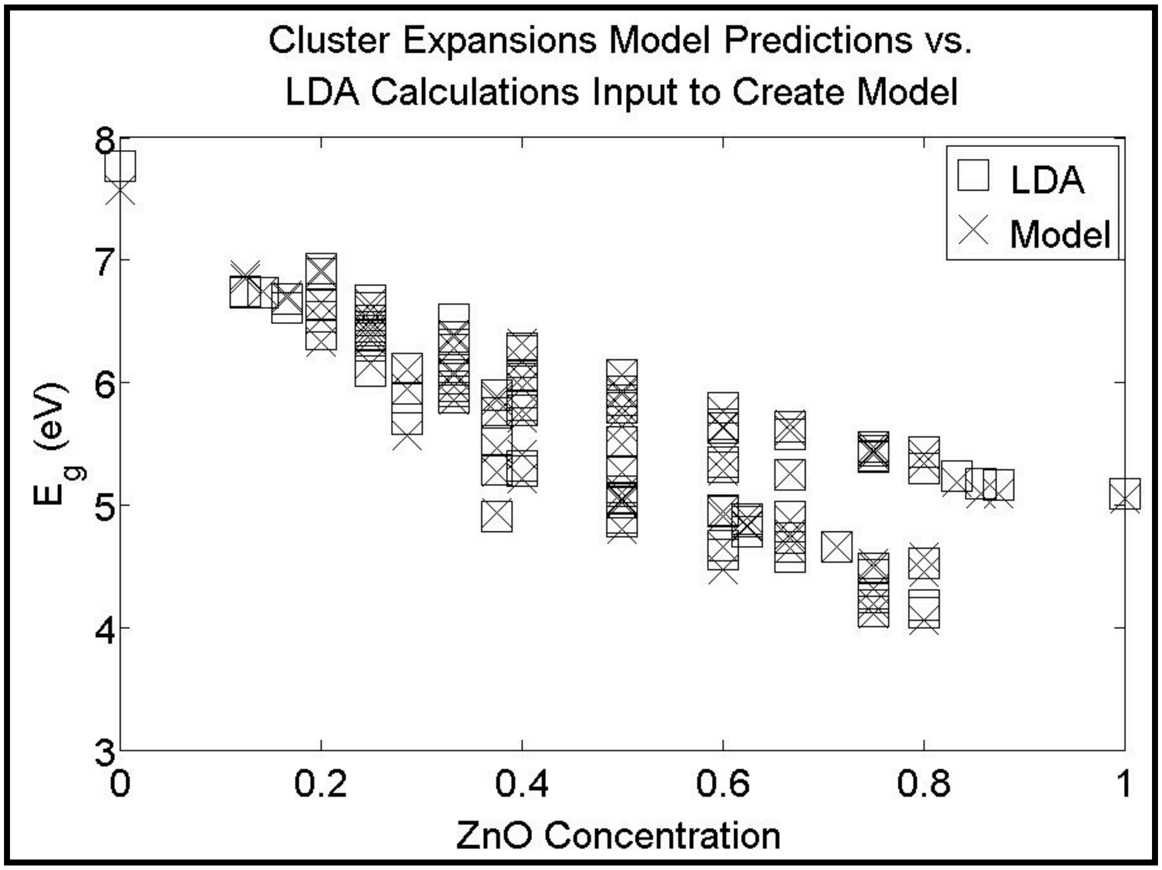


Figure 4: Cluster expansion predictions compared to first principles calculations.

4.2 Total Model Predictions

The cluster expansion model was now used to predict the band gaps for all alloy crystal structures that contain 13 or less cations per primitive cell. This represents the band gaps of over 16,000 structures computed in minutes shown in Figure 5. Here the each small circle represents the predicted band gap for a unique structure while the solid line is the predicted band gap for the random alloy.

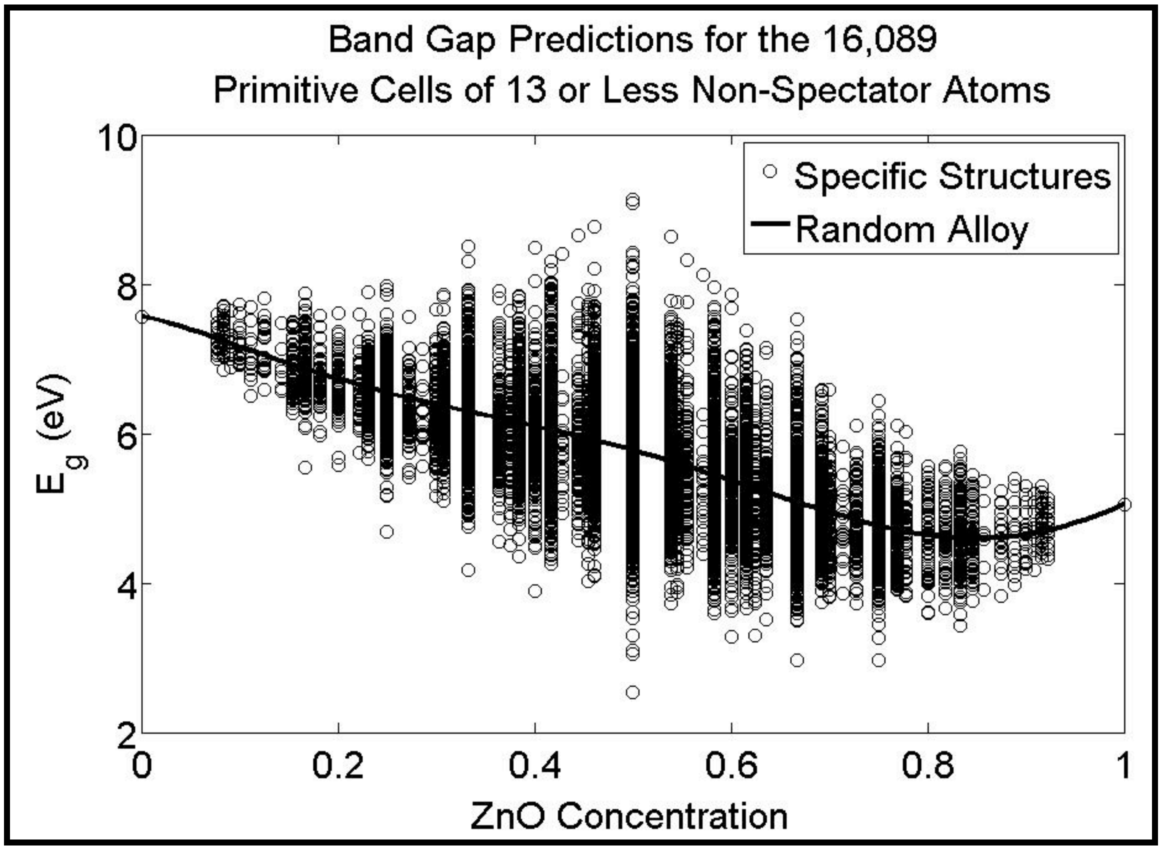


Figure 5: Predicted band gaps for over 16,000 structures of rocksalt MgO-ZnO.

There are three significant results that can be inferred from this busy plot. Most importantly, at many concentrations there is a significant spread of band gaps, which indicates that band gap engineering over a wide range of energies, as much as 6 eV, is possible and is only limited by the technological construction methods at hand.

Second, even if advances in construction are many years in the future, the random alloy can certainly be produced with current technology. This plot shows that the random alloy spans almost 2 eV so that simple band gap engineering can still take place over a large range.

Lastly, there appear to be a large number of wide band gap alloy structures with gaps much larger than that of pure MgO.

Since the rocksalt MgO-ZnO alloy has been observed for ZnO concentrations less than or equal to 50%, only predictions in this concentration range were investigated (Figure 6). As the concentration approaches 50%, the combinatorial number of unique arrangements of 13 or less cations per primitive cell increases so that the number of unique structures increases over this range.

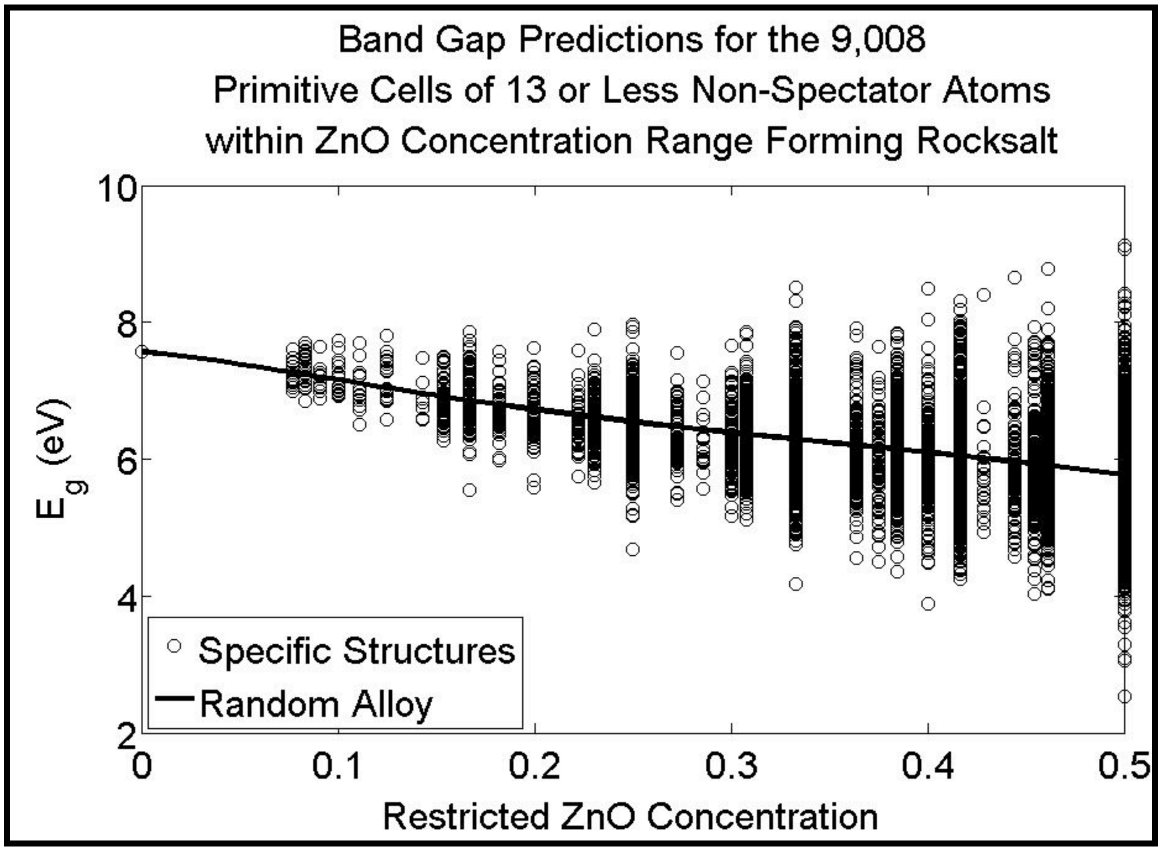


Figure 6: Predicted band gaps over 0-50% ZnO concentration range.

Finally, Figure 7 shows a reduced set of predicted band gaps, those for structures with 8 or less cations per primitive cell, along with the experimentally measured values. The random alloy predicted band gaps nicely approximates the experimentally measured values. The random alloy was most likely synthesized in each laboratory experiment though only determination of the lattice type (wurtzite or rocksalt) was made after synthesis.

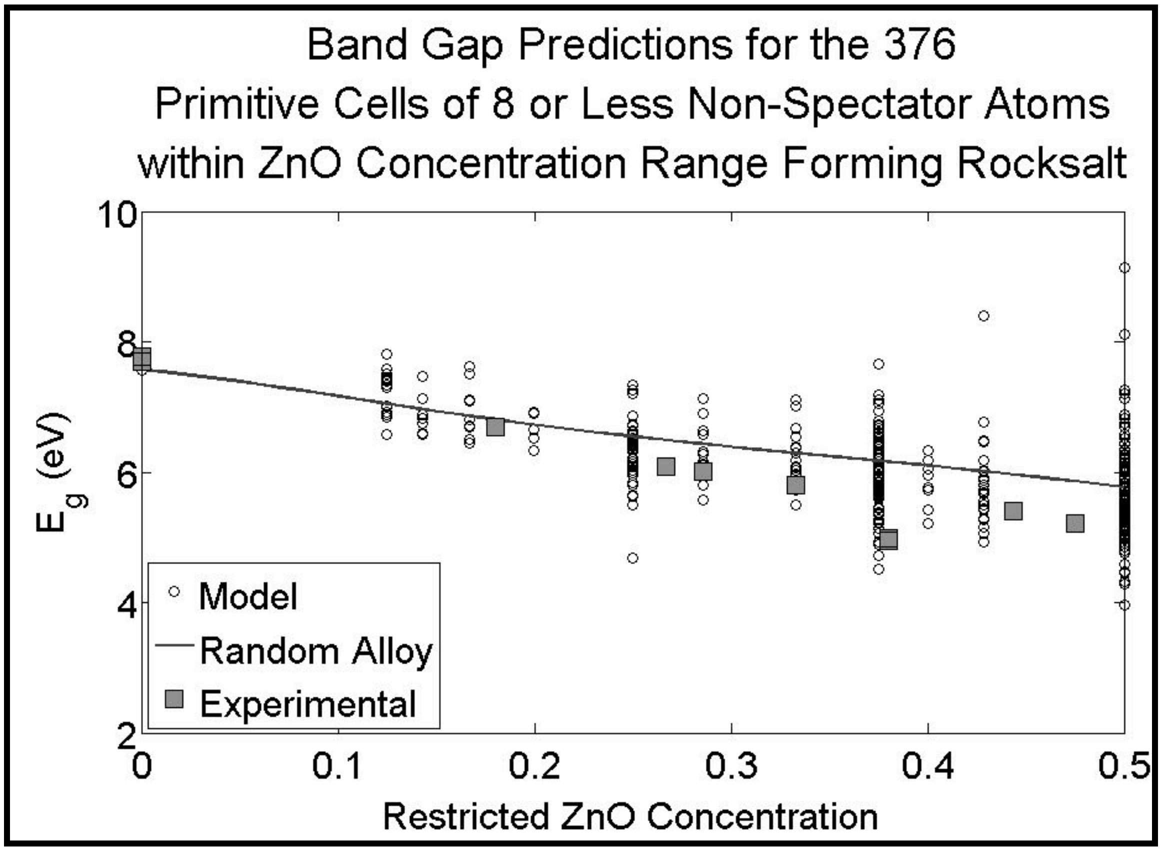


Figure 7: Predicted band gaps over 0-50% ZnO concentration range.

4.3 Very Wide Band Gap Alloy Structures

The largest band gap predicted by the cluster expansion is 9.13 eV. The crystal structure that yields this gap is shown in Figure 8. Here the white spheres represent Mg and the shaded spheres Zn. The interstitial spectator oxygen atoms are not shown.

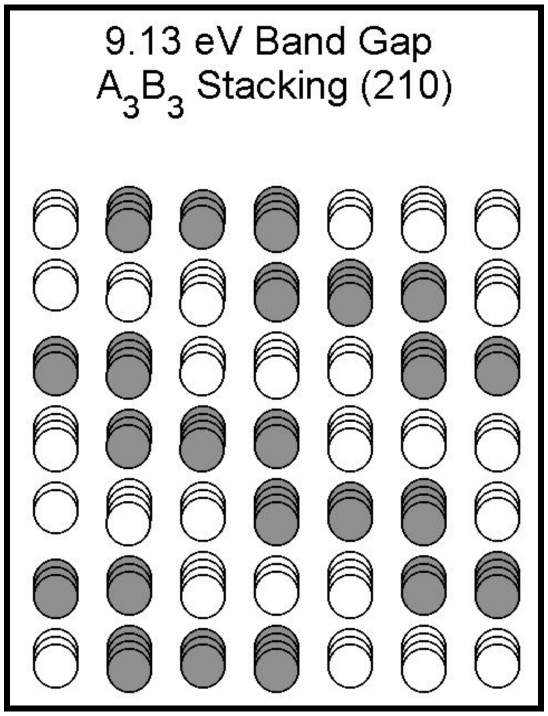


Figure 8: MgO-ZnO with 9.13 eV predicted band gap.

This structure is formed by A_3B_3 stacking in the (210) direction of the rocksalt crystal structure. There is no complete nomenclature at hand to describe over 16,000 unique crystal structures. However, if the structure forms a superlattice, a lattice that can be described by the stacking of two dimensional homopolar layers, then the heading of Figure 8 can uniquely determine the crystal structure. The terminology for stacking sequences, A_3B_3 in this case, is widely used. The A is understood to represent a layer of

Zn cations while the B a layer of Mg atoms. For this structure, three layers of Zn atoms of followed by three layers of Mg atoms, and this is repeated. The stacking direction can be visualized as the sum of three vectors on a face-centered cubic (FCC) lattice. Finally, the stacking order subscripts also give the stoichiometry of the alloy, MgO_3ZnO_3, and hence a 50% concentration with a 6 cation primitive cell.

Several other structures that yield very wide band gap predictions also stack in the (210) direction (Figures 9-12). Of all simplest crystal structures with 8 cations or less per primitive cell, this direction consistently yields the largest band gaps.

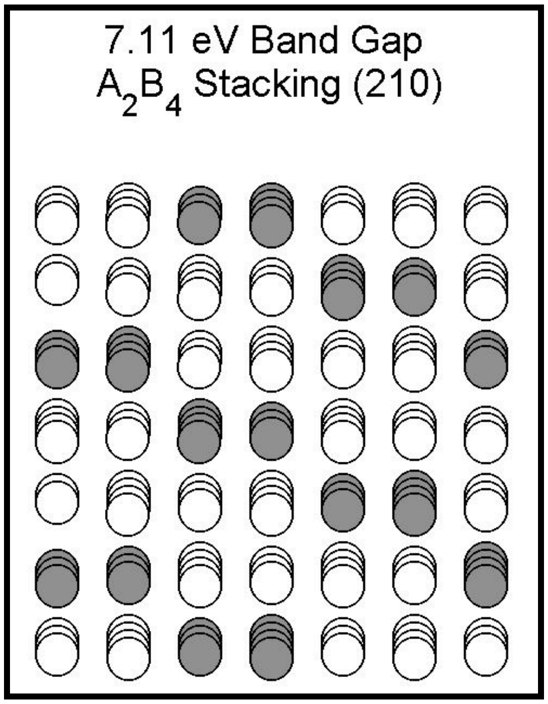


Figure 9: (210) stacking yielding a 7.11 eV predicted band gap.

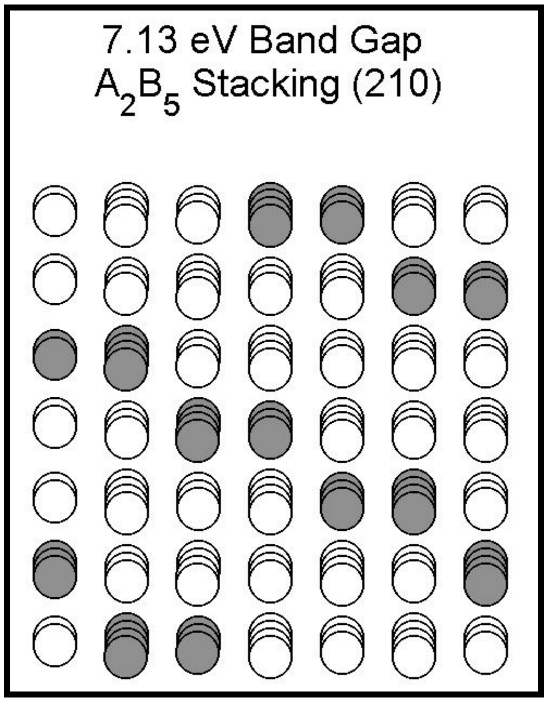


Figure 10: (210) stacking yielding a 7.13 eV predicted band gap.

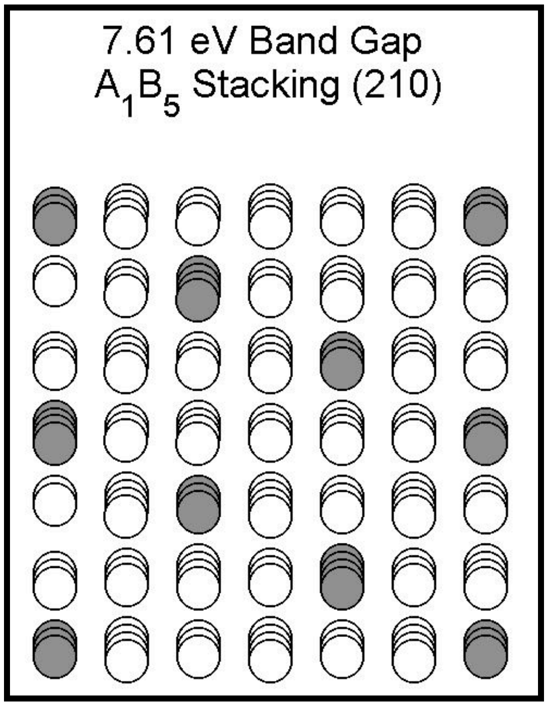


Figure 11: (210) stacking yielding a 7.61 eV predicted band gap.

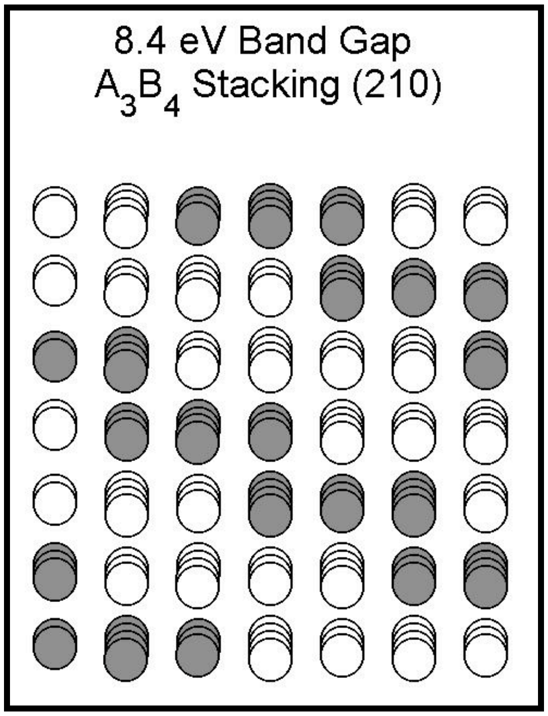


Figure 12: (210) stacking yielding a 8.40 eV predicted band gap.

Of course there were many large superlattice arrangements (crystal structures formed by two dimensional layers of one atom type), but these are well beyond the limitations of current atomic engineering technology (Figures 13 and 14).

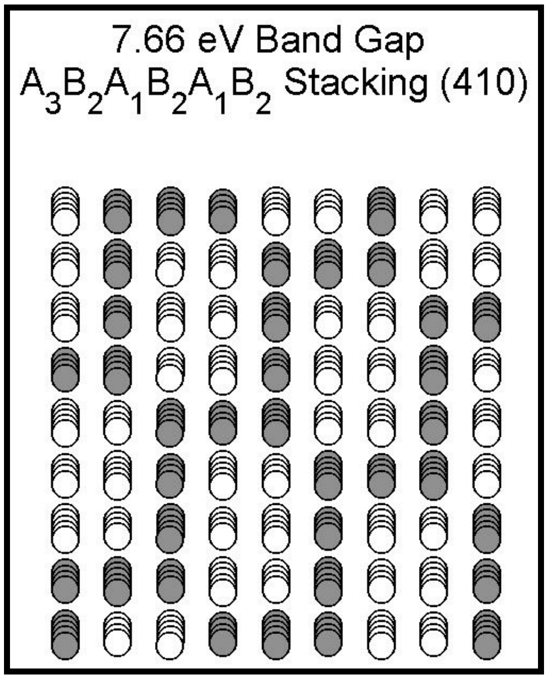


Figure 13: Complicated stacking yielding a 7.66 eV predicted band gap.

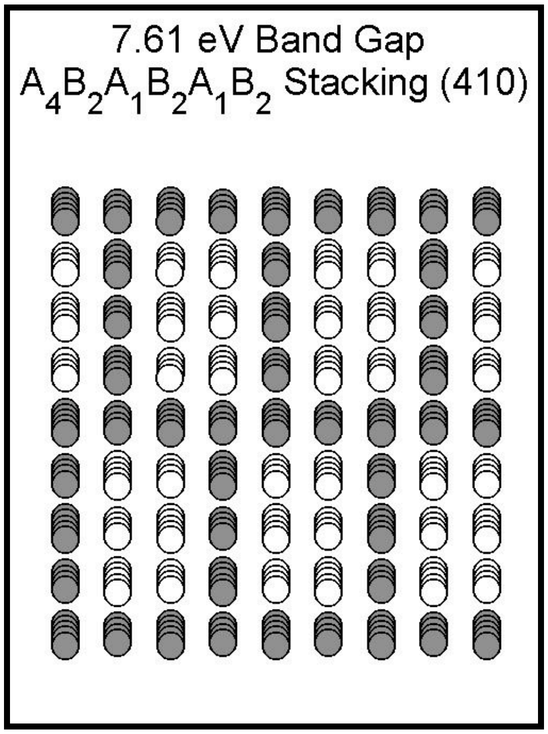


Figure 14: Complicated stacking yielding a 7.61 eV predicted band gap.

Finally, there are superstructures that are not superlattices in that there is no stacking arrangement of atoms that can also provide large band gaps. These would also be extremely difficult to synthesize (Figure 15).

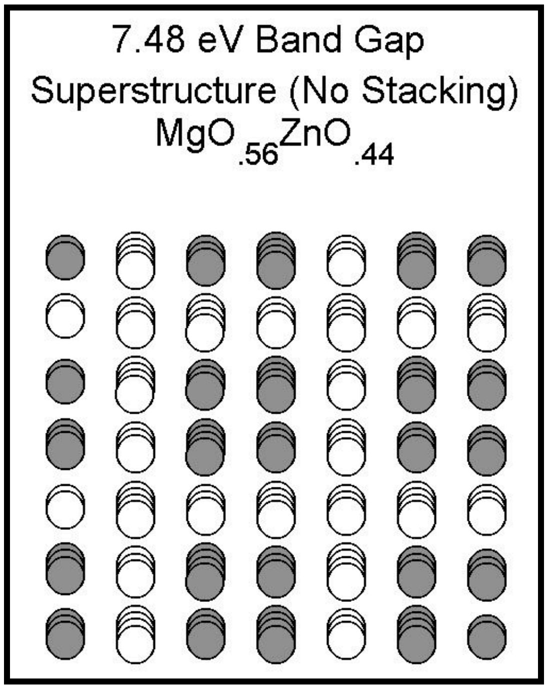


Figure 15: Non-stacking superstructure with 7.48 eV predicted band gap.

4.4 Band Gap Engineering 6.0 eV Alloys

A 6.0 eV band gap represents a photonic wavelength of 207 nm, placing it at the high energy boundary of the UV-C spectrum (200-280 nm). Semiconductor devices operating at this wavelength would have astronometric and defense applications. The crystal structures that are predicted to deliver this band gap were investigated.

Most synthesized rocksalt structures currently grow in the (100) stacking direction. Figure 16 show the only crystal structure with a relatively simple stacking arrangement that stacks in this direction with a predicted band gap of ~6 eV.

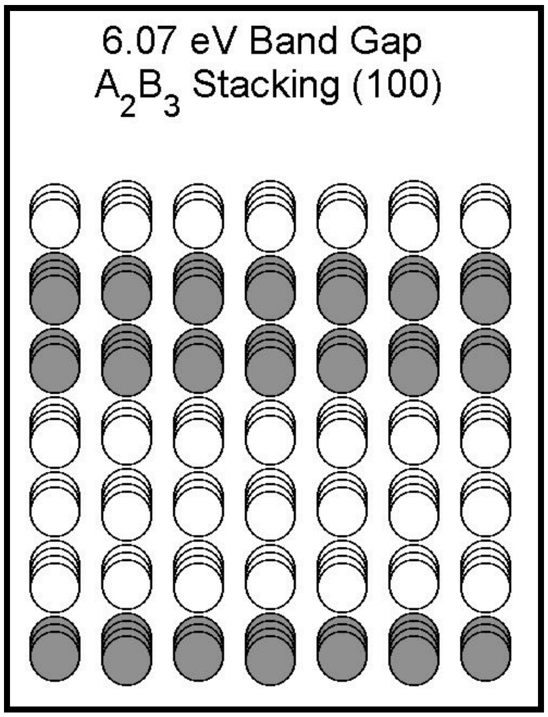


Figure 16: Engineering a 6 eV band gap in the (100) direction.

One laboratory has reported growth in the (111) direction.[10] Likewise, only one simple stacking arrangement in this direction is predicted to yield a 6 eV band gap (Figure 17).

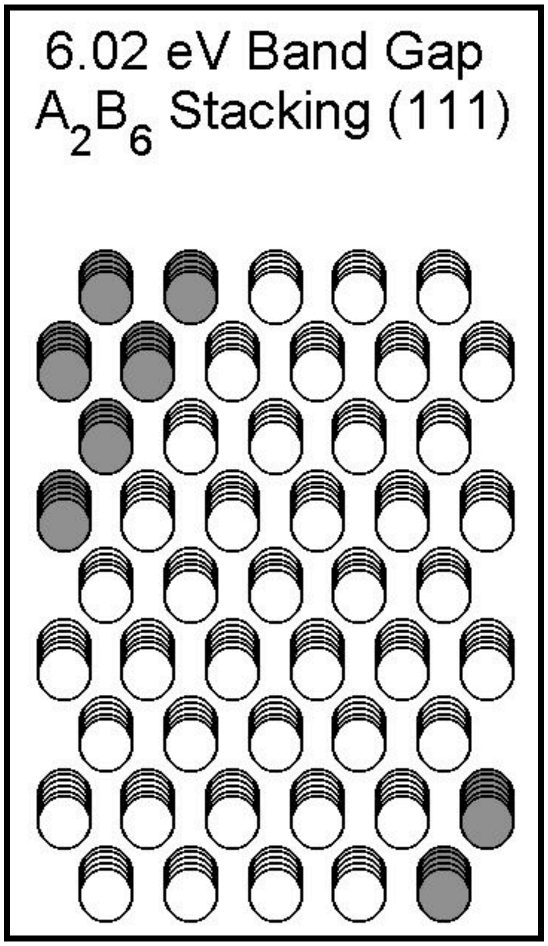


Figure 17: Engineering a 6 eV band gap in the (111) direction.

Figure 18 shows the only simple stacking sequence growing in the (110) direction to give a 6 eV band gap.

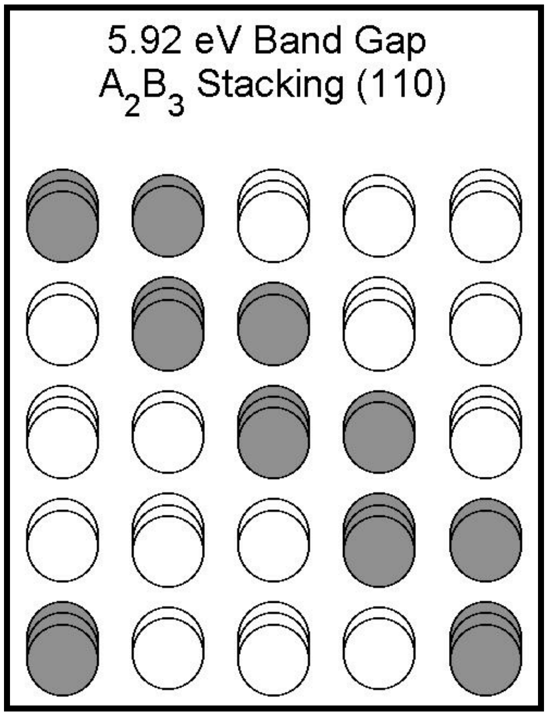


Figure 18: Engineering a 6 eV band gap in the (110) direction.

Two candidates for engineering a 6 eV band gap were found stacking in the (311) direction (Figures 19 and 20), three more were found stacking in the (331) direction (Figures 21-23) and finally one was found to stack in the (531) direction (Figure 24).

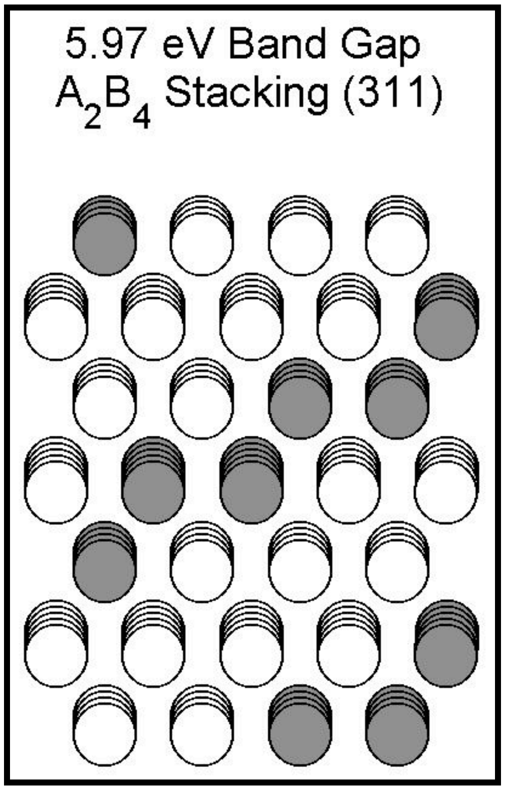


Figure 19: Engineering a 6 eV band gap in the (311) direction.

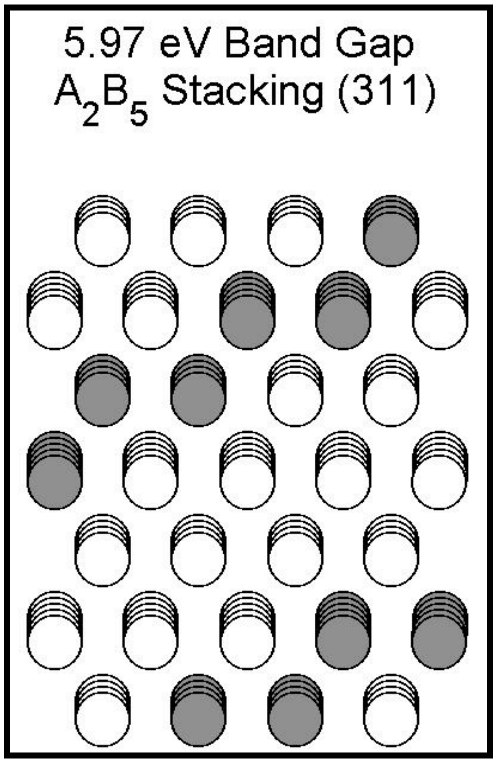


Figure 20: Engineering a 6 eV band gap in the (311) direction.

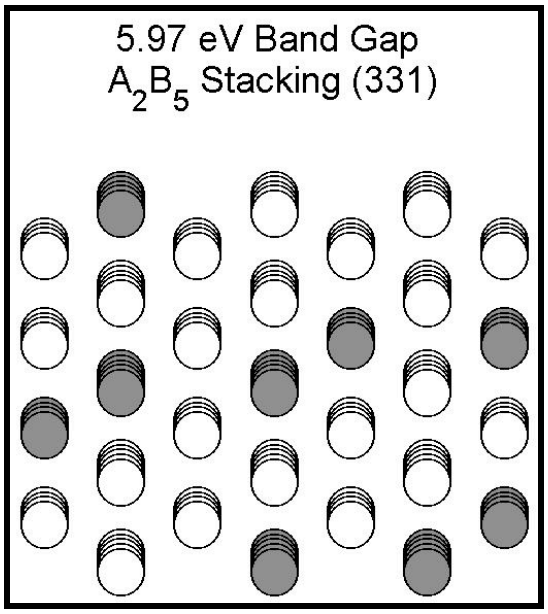


Figure 21: Engineering a 6 eV band gap in the (331) direction.

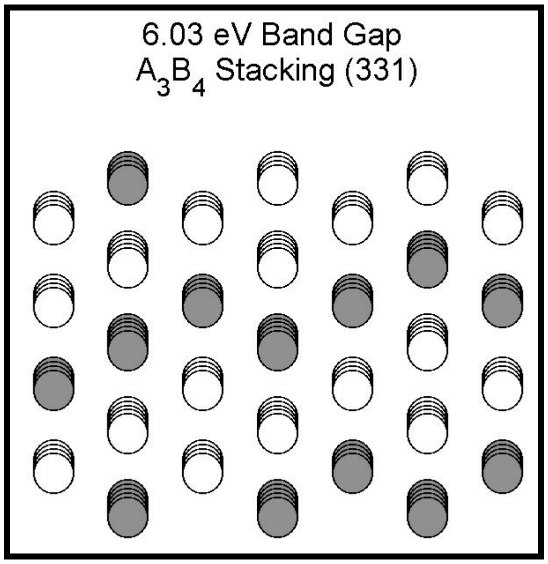


Figure 22: Engineering a 6 eV band gap in the (331) direction.

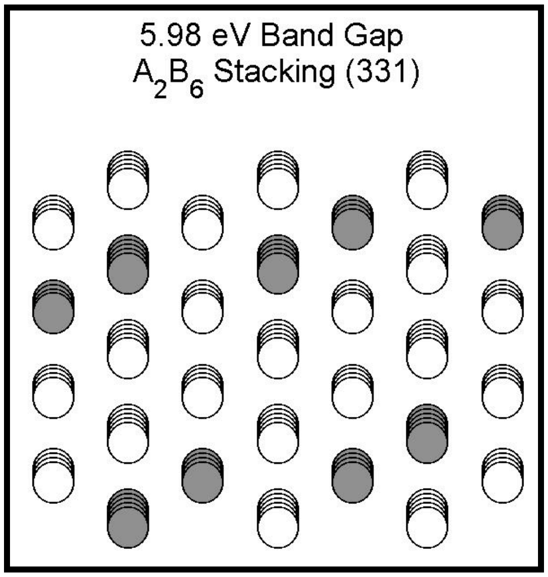


Figure 23: Engineering a 6 eV band gap in the (331) direction.

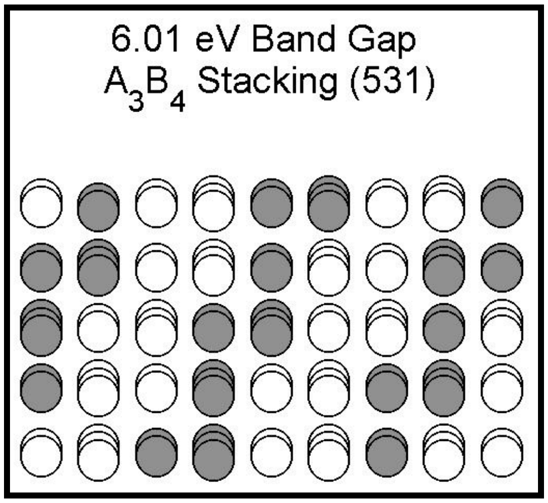
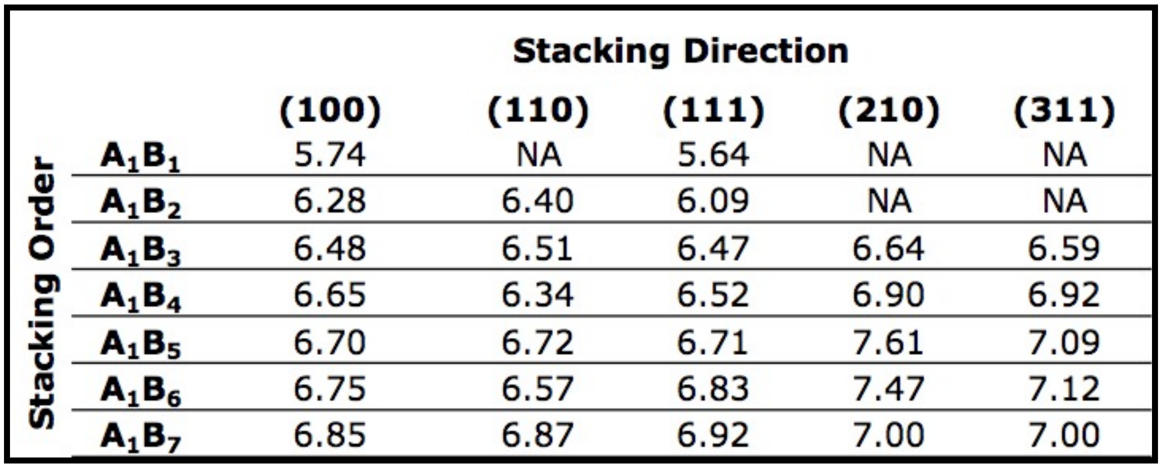


Figure 24: Engineering a 6 eV band gap in the (531) direction.

These eight structures provide an exhaustive list of simple stacking sequences that are predicted to engineer a 6 eV band gap alloy.

4.5 Directional Dependence of Stacking

With the predicted band gaps of over 16,000 unique structures, there is an enormous database of data through which to examine for relationships in the data. One investigation that was performed was examining the band gap trends for A_1B_x stacking in several simple stacking directions where x implies that the MgO concentration is increasing. The predicted band gaps for these specific crystal structures is given in Table 1.



	Stacking Direction				
Stacking Order	(100)	(110)	(111)	(210)	(311)
A_1B_1	5.74	NA	5.64	NA	NA
A_1B_2	6.28	6.40	6.09	NA	NA
A_1B_3	6.48	6.51	6.47	6.64	6.59
A_1B_4	6.65	6.34	6.52	6.90	6.92
A_1B_5	6.70	6.72	6.71	7.61	7.09
A_1B_6	6.75	6.57	6.83	7.47	7.12
A_1B_7	6.85	6.87	6.92	7.00	7.00

Table 1: Predicted band gaps as a function of stacking order and direction.

For any given stacking order, crystal structures all possess the same stoichiometry, and there predicted band gap are similar. All directions seem to proceed toward a common band gap value, though not all do so monotonically. Further, the limiting value appears to be well below the band gap energy for pure MgO, ~7.8 eV. This indicates that the valence band must contain a fair amount of charge localization on the Zn atoms' d or s orbitals.

Stacking direction (210) that was shown earlier to be a direction for very wide band gaps is nearly always the largest for a given stacking order. However, direction (210) does not show the same general trend of increasing band gap with increasing MgO concentration as the others, but falls rapidly after the 83% MgO concentration level (A_1B_5). Though no explanation of this effect is given here, stacking in this direction in some way must affect the charge localization on the Zn atoms.

4.6 Wide Band Gap Recipes

Finally, Table 2 provides a listing of the smallest primitive cells and their band gap energies as a list of recipes for producing several band gap energies.

Superlattice Direction	Stacking Oder	ZnO Concentration	Predicted Band Gap
[1,1,1]	A2B2	0.50	5.0
[3,1,1]	A2B2	0.50	5.3
[1,1,1]	A1B1	0.50	5.6
[1,0,0]	A1B1	0.50	5.7
[1,1,0]	A2B2	0.50	5.8
[1,0,0]	A2B2	0.50	5.9
[2,1,0]	A2B2	0.50	6.1
[1,1,1]	A1B2	0.33	6.1
[1,1,1]	A1B3	0.25	6.5
[1,0,0]	A1B2	0.33	6.3
[1,0,0]	A1B3	0.25	6.5
[1,1,0]	A1B3	0.25	6.5
[1,1,1]	A1B4	0.20	6.5
[1,1,0]	A1B2	0.33	6.4
[1,1,0]	A1B4	0.20	6.3
[2,1,0]	A1B3	0.25	6.6
[3,1,1]	A1B3	0.25	6.6
[1,0,0]	A1B4	0.20	6.6
[3,1,1]	A1B4	0.20	6.9
[2,1,0]	A1B4	0.20	6.9

Table 2: Smallest primitive cell crystal structures band gap predictions.

Chapter 5

Summary of Research

Using first principles calculations and guided by experimental measurements, a cluster expansion of the $MgO_{1-x}ZnO_x$ wide band gap semiconducting alloy was constructed for the rocksalt crystal structure. This model functional for the band gap energy was a function of the geometry of the underlying FCC lattice of cation sites only so that rapid band gap predictions could be made for a myriad of unique crystal structures.

The resultant model predicted a large spread in achievable band gaps for many different ZnO concentrations and a smaller, though still significant, spread in the predicted random alloy band gaps over the range of ZnO concentrations. Several structures were investigated that yielded particularly wide band gaps, and the (210) directional superlattice stacking was seen to provide the widest band gaps for the simplest primitive cells.

The possible engineering of a 6 eV band gap alloy of MgO-ZnO was examined, and it was determined that there were eight reasonably simple stacking arrangements that would achieve this target. These eight candidates were found to be at several ZnO concentrations and in different superlattice stacking directions so that no perceivable similarities existed between the candidate crystal structures.

Finally, several stacking directions were investigated for the A_1B_x, Mg rich stacking order and it was determined from this analysis that charge localization on the Zn cation occurred in the valence band of the alloys.

References

[1] G. Burns, *Solid State Physics*, (Academic Press, Inc., Orlando, 1985), p. 252.

[2] C. R. Nave, *HyperPhysics*, (Georgia State University, Department of Physics and Astronomy), http://hyperphysics.phy-astr.gsu.edu/hbase/hframe.html.

[3] M. Sanati, G. L. W. Hart and A. Zunger, Phys. Rev. B, **68**, 155210 (2003).

[4] S. Choopun, R. D. Vispute, W. Yang, R. P. Sharma, and T. Venkatesan, App. Phys. Lett., 80, **9**, 1529 (2002).

[5] J. Narayan, A. K. Sharma, A. Kvit, C. Jin, J.F. Muth, and O. W. Holland, Solid St. Comm. 121, **9**, 9 (2002).

[6] A. Othomo, M. Kawasaki, T. Koida, K. Masubuchi, H. Koinuma, Y. Sakurai, Y. Yoshida, T. Yasuda, Y. Segawa, App. Phys. Lett. 72, **19**, 2466 (1998).

[7] W. Yang, R. D. Vispute, S. Choopun, R. P. Sharma, T. Venkatesan, and H. Shen, App. Phys. Lett., 78, **18**, 2787 (2001).

[8] W. Yang, S. S. Hullavarad, B. Nagaraj, I. Takeuchi, R. P. Sharma, T. Venkatesan, R. D. Vispute, and H. Shen, App. Phys. Lett., 82, **20**, 3424 (2003).

[9] T. Minemoto, T. Negami, S. Nishiwaki, H. Takakura, and Y. Hamakawa, Thin Solid Films **372**, 173 (2000).

[10] G-J. Fang, D. Li, B-L. Yao and X. Zhao, J. Cryst. Growth **258**, 310 (2003).

[11] E. Kaxiras, *Atomic and Electronic Structure of Solids*, (Cambridge University Press, Cambridge, 2003) p. 59.

[12] M. P. Das, in *Lectures on Methods of Electronic Structure Calculation*, edited by V. Kumar, O. K. Andersen and A. Mookerjee (World Scientific Publishing, Singapore, 1994), p. 1.

[13] R. Martin, *Electronic Structure: Basic Theory and Practical Methods*, (Cambridge University Press, Cambridge, 2004) p. 157.

[14] K. Laasonen, in *Lectures on Methods of Electronic Structure Calculation*, edited by V. Kumar, O. K. Andersen and A. Mookerjee (World Scientific Publishing, Singapore, 1994), p. 371.

[15] D. Laks, L. Ferreira, S. Froyen, and A. Zunger, Phys. Rev. B, 46 **19**, 12587 (1992).

[16] V, Blum, G. L. W. Hart, M. J. Walorski, and A. Zunger, Phys. Rev. B **72**, 165113 (2005).

[17] A. van der Walle and G. Ceder, J. Phase Equil. **23**, 348 (2002).

[18] K. Baumann, Trends Anal. Chem. **22**, 395 (2003).

[19] J. Shao, J. Am. Stat. Assoc. **88**, 486 (1993).

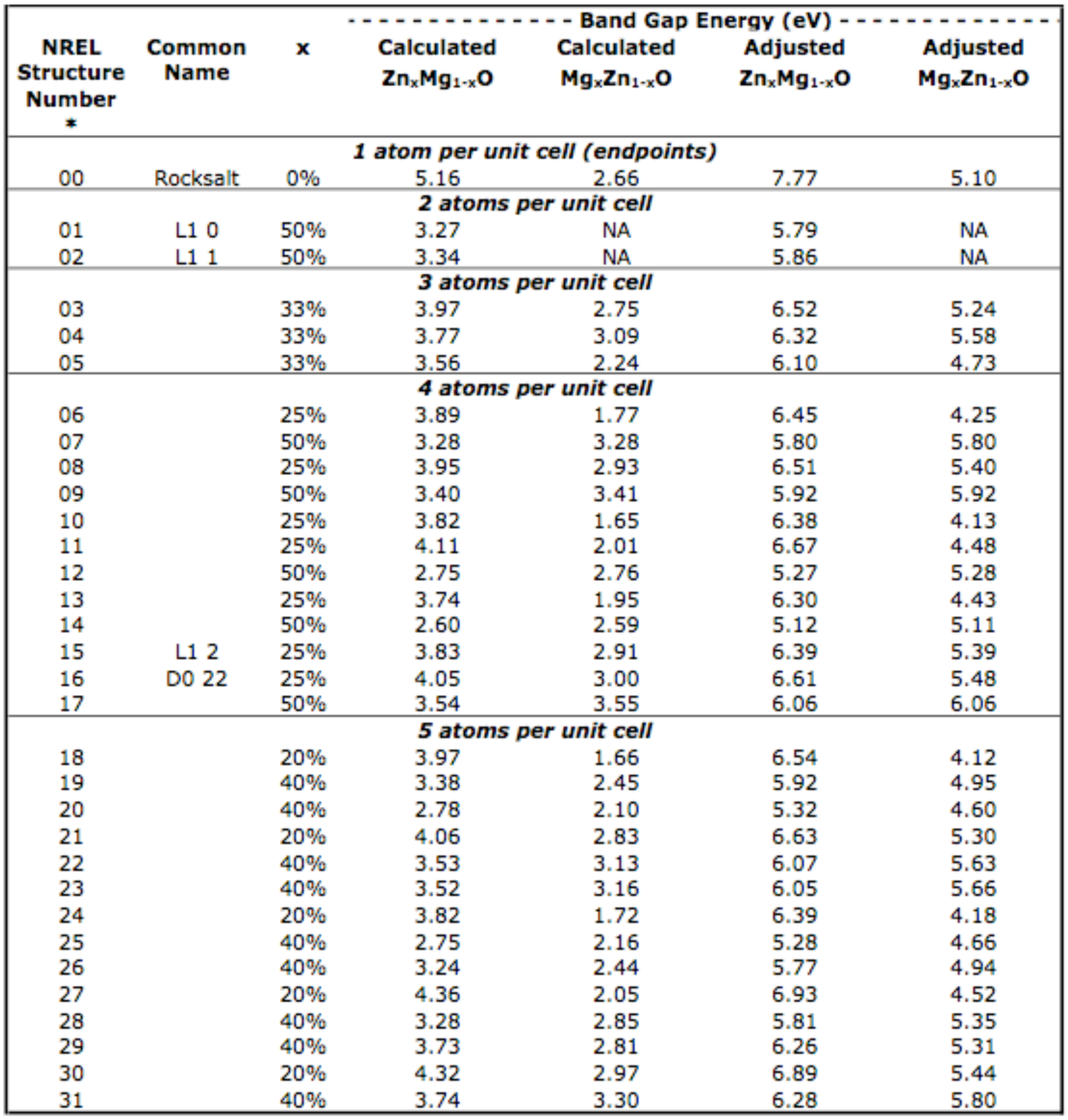
[20] G. L. W. Hart, V. Blum, M. J. Walorski, and A. Zunger, Nature Materials **4**, 391 (2005).

[21] R. Magri and A.Zunger, Phys. Rev. B **16**, 8672 (1991).

[22] W. R. L. Lambrecht, S. Limpinjumnong and B. Segall, MRS Internet J. Nitride Semicond. Res. **4S1**, G6.8 (1999).

Appendices

A.1. First Principles Data



			Band Gap Energy (eV)			
NREL Structure Number *	Common Name	x	Calculated $Zn_xMg_{1-x}O$	Calculated $Mg_xZn_{1-x}O$	Adjusted $Zn_xMg_{1-x}O$	Adjusted $Mg_xZn_{1-x}O$
			1 atom per unit cell (endpoints)			
00	Rocksalt	0%	5.16	2.66	7.77	5.10
			2 atoms per unit cell			
01	L1 0	50%	3.27	NA	5.79	NA
02	L1 1	50%	3.34	NA	5.86	NA
			3 atoms per unit cell			
03		33%	3.97	2.75	6.52	5.24
04		33%	3.77	3.09	6.32	5.58
05		33%	3.56	2.24	6.10	4.73
			4 atoms per unit cell			
06		25%	3.89	1.77	6.45	4.25
07		50%	3.28	3.28	5.80	5.80
08		25%	3.95	2.93	6.51	5.40
09		50%	3.40	3.41	5.92	5.92
10		25%	3.82	1.65	6.38	4.13
11		25%	4.11	2.01	6.67	4.48
12		50%	2.75	2.76	5.27	5.28
13		25%	3.74	1.95	6.30	4.43
14		50%	2.60	2.59	5.12	5.11
15	L1 2	25%	3.83	2.91	6.39	5.39
16	D0 22	25%	4.05	3.00	6.61	5.48
17		50%	3.54	3.55	6.06	6.06
			5 atoms per unit cell			
18		20%	3.97	1.66	6.54	4.12
19		40%	3.38	2.45	5.92	4.95
20		40%	2.78	2.10	5.32	4.60
21		20%	4.06	2.83	6.63	5.30
22		40%	3.53	3.13	6.07	5.63
23		40%	3.52	3.16	6.05	5.66
24		20%	3.82	1.72	6.39	4.18
25		40%	2.75	2.16	5.28	4.66
26		40%	3.24	2.44	5.77	4.94
27		20%	4.36	2.05	6.93	4.52
28		40%	3.28	2.85	5.81	5.35
29		40%	3.73	2.81	6.26	5.31
30		20%	4.32	2.97	6.89	5.44
31		40%	3.74	3.30	6.28	5.80

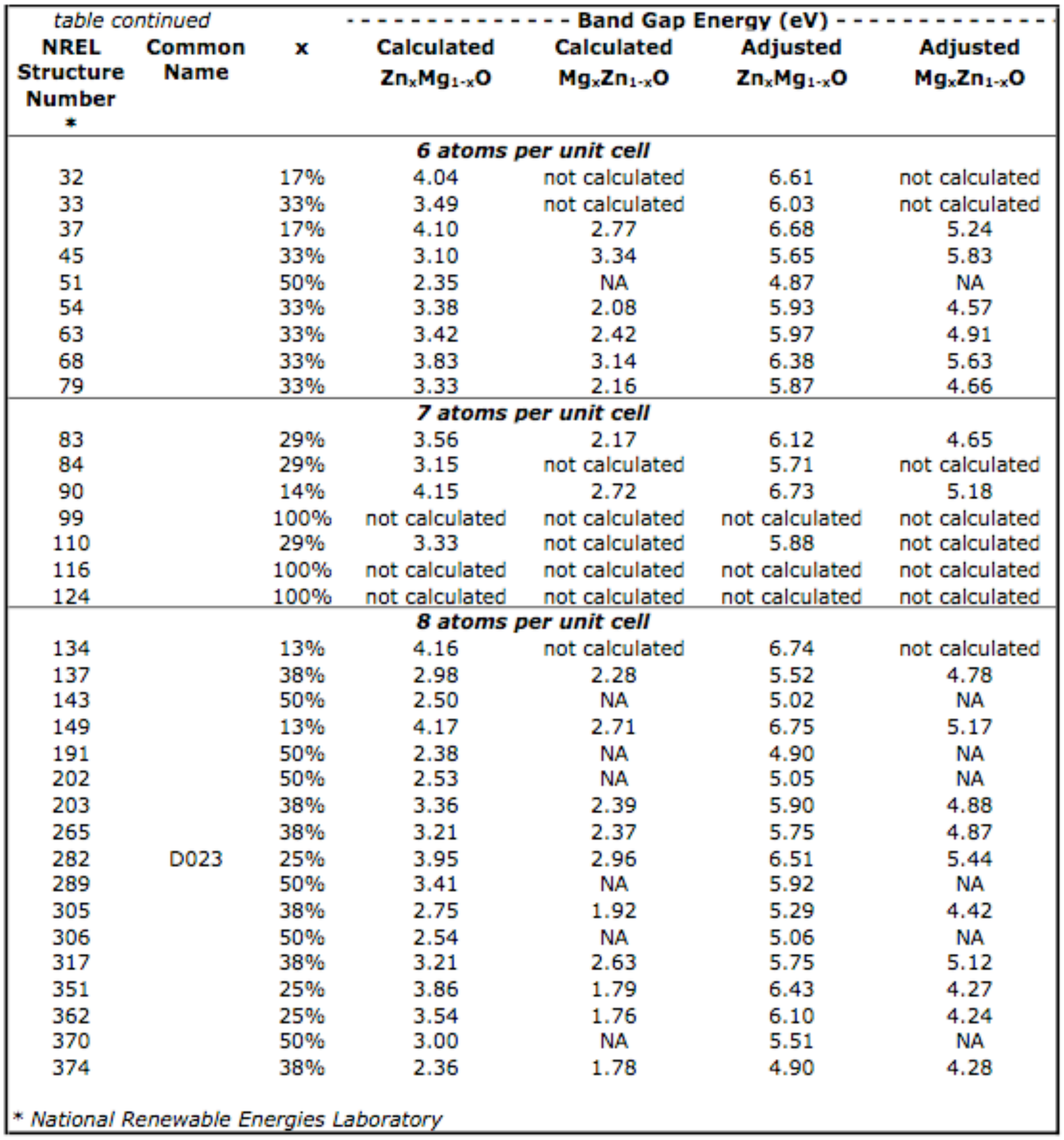


table continued			- - - - - - - - - - - - - - Band Gap Energy (eV)			- - - - - - - - - - - - - -
NREL Structure Number *	**Common Name**	**x**	**Calculated $Zn_xMg_{1-x}O$**	**Calculated $Mg_xZn_{1-x}O$**	**Adjusted $Zn_xMg_{1-x}O$**	**Adjusted $Mg_xZn_{1-x}O$**
			6 atoms per unit cell			
32		17%	4.04	not calculated	6.61	not calculated
33		33%	3.49	not calculated	6.03	not calculated
37		17%	4.10	2.77	6.68	5.24
45		33%	3.10	3.34	5.65	5.83
51		50%	2.35	NA	4.87	NA
54		33%	3.38	2.08	5.93	4.57
63		33%	3.42	2.42	5.97	4.91
68		33%	3.83	3.14	6.38	5.63
79		33%	3.33	2.16	5.87	4.66
			7 atoms per unit cell			
83		29%	3.56	2.17	6.12	4.65
84		29%	3.15	not calculated	5.71	not calculated
90		14%	4.15	2.72	6.73	5.18
99		100%	not calculated	not calculated	not calculated	not calculated
110		29%	3.33	not calculated	5.88	not calculated
116		100%	not calculated	not calculated	not calculated	not calculated
124		100%	not calculated	not calculated	not calculated	not calculated
			8 atoms per unit cell			
134		13%	4.16	not calculated	6.74	not calculated
137		38%	2.98	2.28	5.52	4.78
143		50%	2.50	NA	5.02	NA
149		13%	4.17	2.71	6.75	5.17
191		50%	2.38	NA	4.90	NA
202		50%	2.53	NA	5.05	NA
203		38%	3.36	2.39	5.90	4.88
265		38%	3.21	2.37	5.75	4.87
282	D023	25%	3.95	2.96	6.51	5.44
289		50%	3.41	NA	5.92	NA
305		38%	2.75	1.92	5.29	4.42
306		50%	2.54	NA	5.06	NA
317		38%	3.21	2.63	5.75	5.12
351		25%	3.86	1.79	6.43	4.27
362		25%	3.54	1.76	6.10	4.24
370		50%	3.00	NA	5.51	NA
374		38%	2.36	1.78	4.90	4.28

** National Renewable Energies Laboratory*

Table 3: LDA calculated and adjusted band gap energies.

A.2. Additional Crystal Structure Results

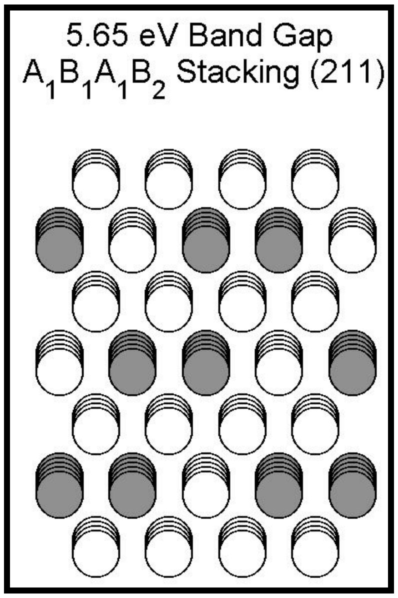


Figure 25: A 5.65 eV band gap alloy.

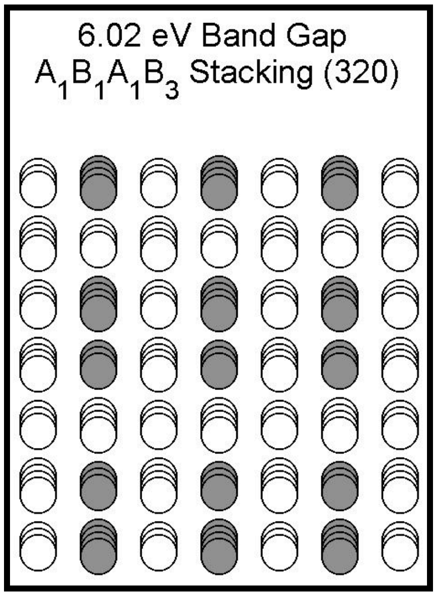


Figure 26: A 6.02 eV band gap alloy.

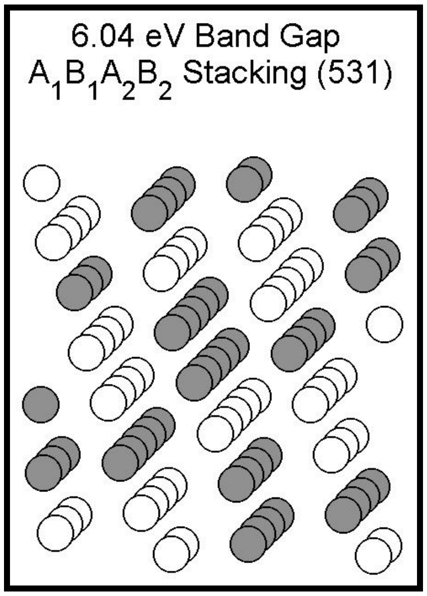


Figure 27: A 6.04 eV band gap alloy.

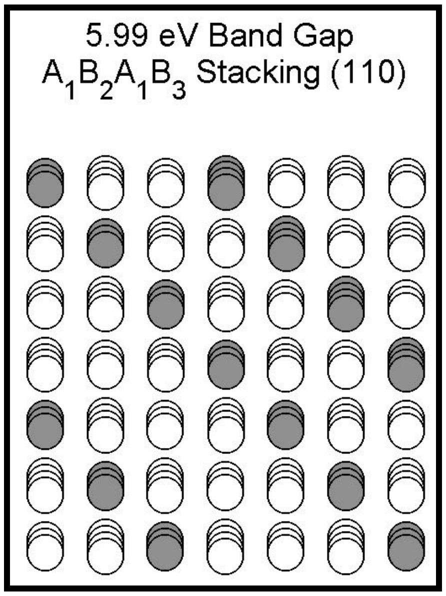


Figure 28: A 5.99 eV band gap alloy.

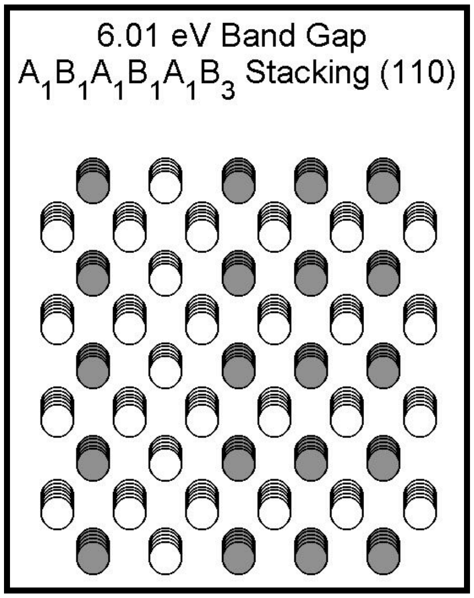


Figure 29: A 6.01 eV band gap alloy.

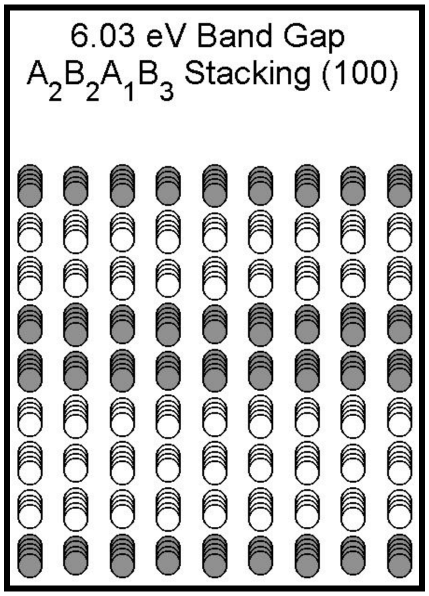


Figure 30: A 6.03 eV band gap alloy.

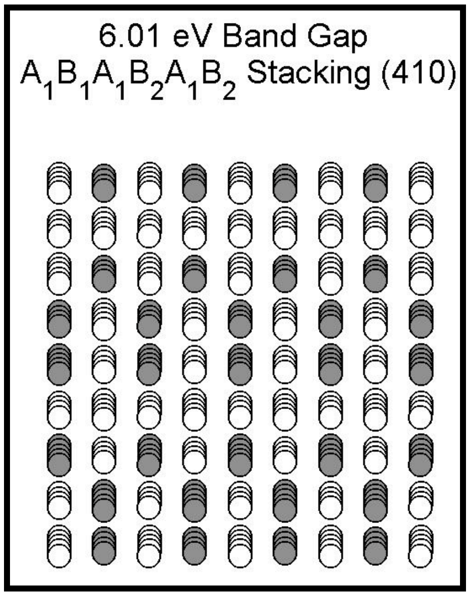


Figure 31: A 6.01 eV band gap alloy.

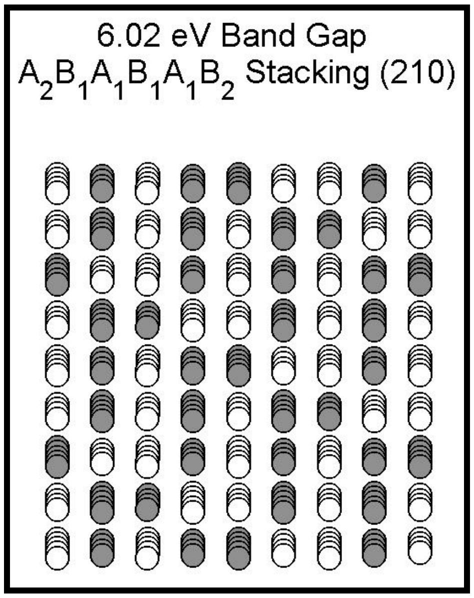


Figure 32: A 6.02 eV band gap alloy.

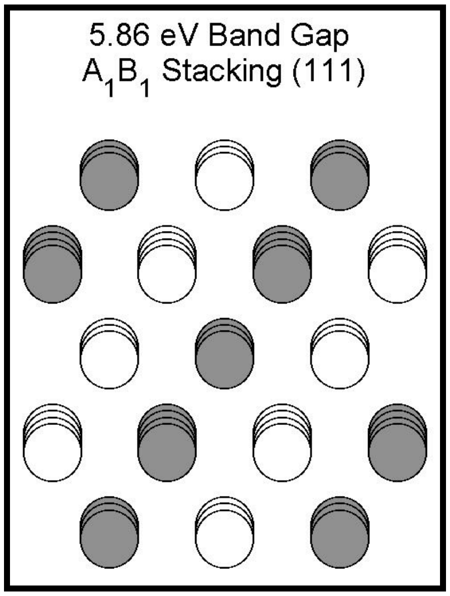


Figure 33: A 5.86 eV band gap alloy.

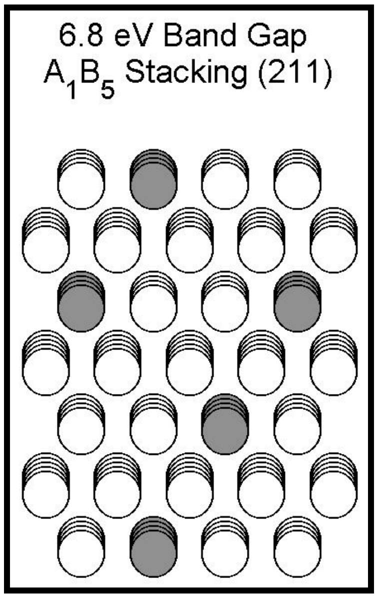


Figure 34: A 6.80 eV band gap alloy.

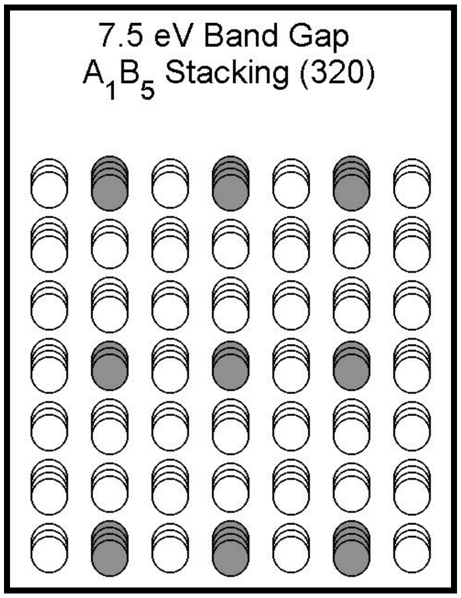


Figure 35: A 7.50 eV band gap alloy.

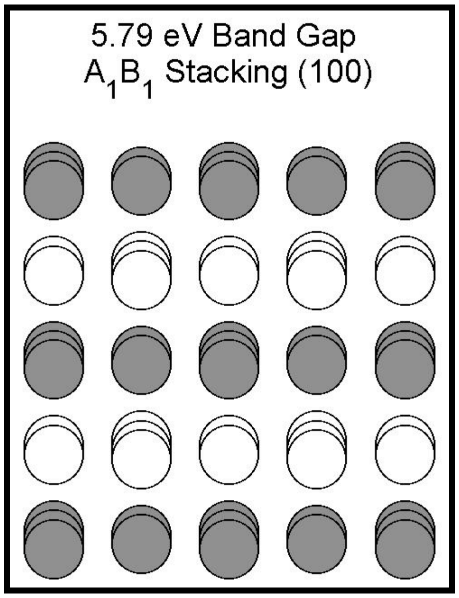


Figure 36: A 5.79 eV band gap alloy.

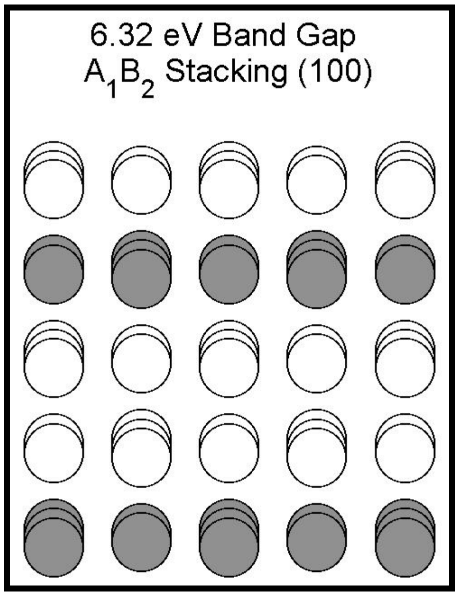


Figure 37: A 6.32 eV band gap alloy.

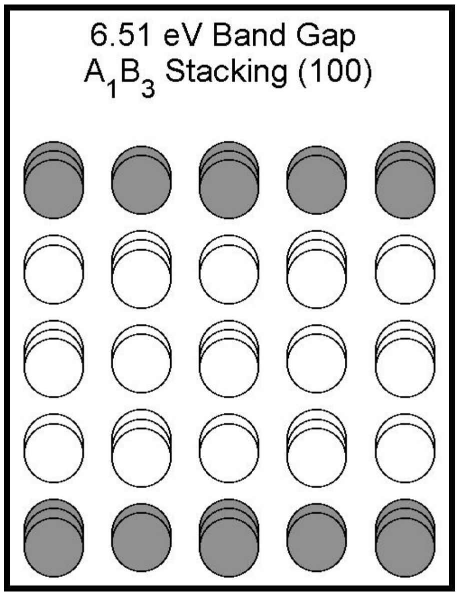


Figure 38: A 6.51 eV band gap alloy.

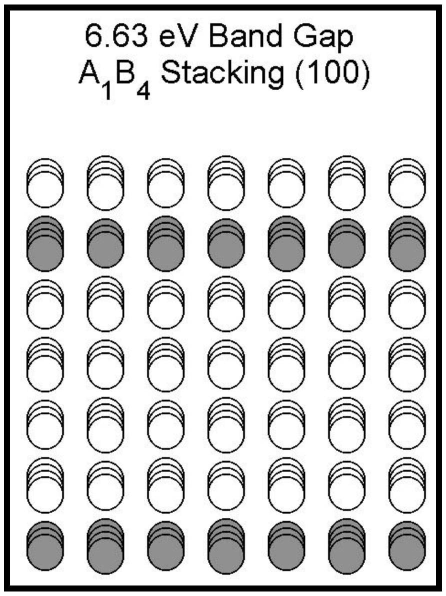


Figure 39: A 6.63 eV band gap alloy.

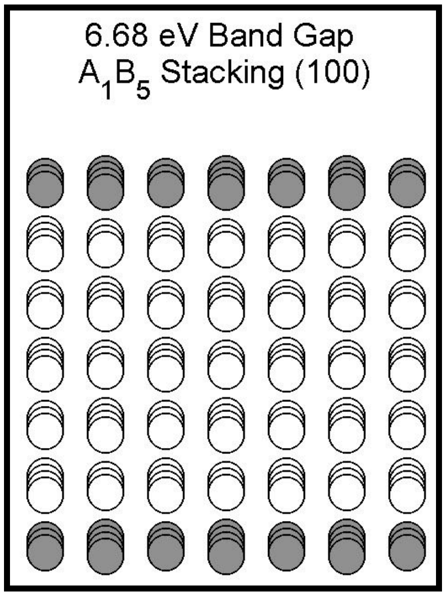


Figure 40: A 6.68 eV band gap alloy.

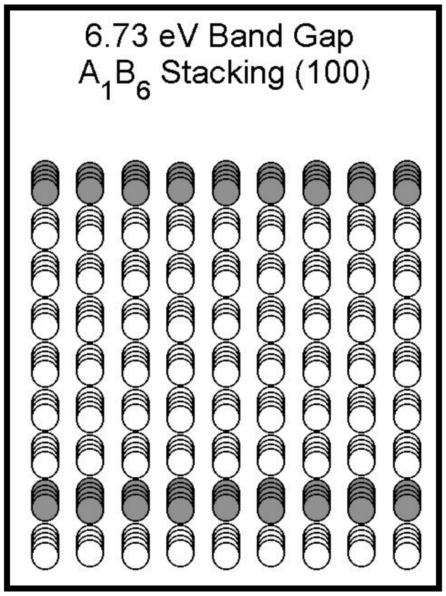


Figure 41: A 6.73 eV band gap alloy.

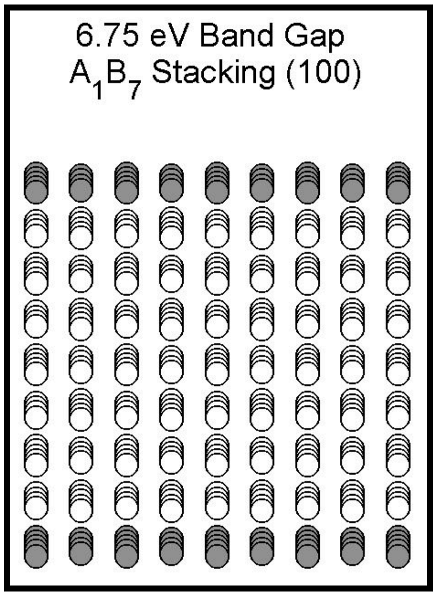


Figure 42: A 6.75 eV band gap alloy.

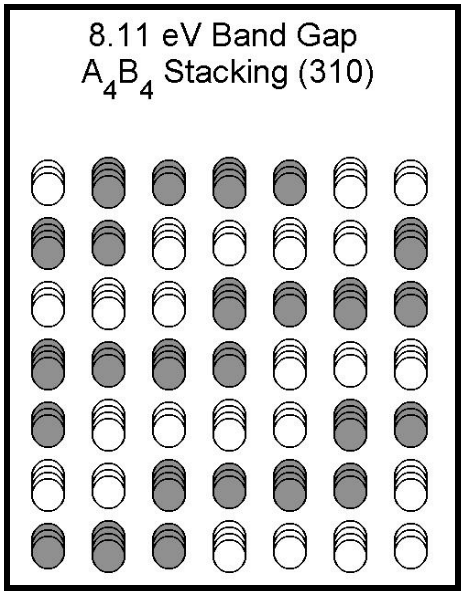


Figure 43: A 8.11 eV band gap alloy.

A.3. Instructions for Cluster Expansion Software

It is important to be able to create new *pi* files in order to incorporate specific multi-body clusters for the genetic algorithm to choose from. In a folder called *list_clusters* are several *.f* files and a very helpful *Readme*. These include

check.f	***compile.list_clusters***	*lattic_sc.f*
clusters.f	*count_equivalent.f*	*list_clusters.log*
clusters.in	*degen_cluster.f*	*list_clusters.pdf*
clusters.out	*figure*	*list_clusters.ps*
compare_clusters.f	*find_cluster.f*	***param.f***
compare_clusters.in	*get_dist.f*	***Readme***
compile.degen_cluster	*lattic_bcc.f*	
compile.find_cluster	*lattic_fcc.f*	

The file *clusters.in* is where to begin. This was originally set to 6 points and 5 max distance. This was piecewise expanded to include 8 vertex points for a larger search space for the genetic algorithm. Now run the bash script *compile.list_clusters* to generate the three executables *list_clusters_bcc.x, list_clusters_fcc.x, list_clusters_sc.x*. Then use of whichever of these that is needed to give the output file, *clusters.out.* This output file **is** the new *jtypes* file, but a corresponding 'PI file' still needs to be created with which to make the predictions.

The *param.f* file was repeatedly adjusted and compiled and run multiple times; each time adding the larger MBITs that were needed. Specifically, first 3-bodies up to 8th nearest neighbors were created. This gave a *cluster.out* that was renamed *jtypes*. Then 4-bodies up to 4th nearest neighbor were calculated, and the 4-body results were copied from the new *cluster.out* to the bottom of the *jtypes*. Finally, up to 8-bodies with 3rd nearest neighbor were run to get the 5 through 8 bodies, and then copied these to *jtypes*. This ended up with 124 MBITs for the GA. Along with the an indeterminate number of pair interactions (~20-30), this gave over 140 interaction types to search through.

Now go to the folder called something like *make_pi*. It should have the following files:

allph00f.in	***compile.fcc***	*makePIs.fcc.f90*
compile.bcc	*makePIs.bcc.f90*	*parameters*

First copy over the *clusters.out* to this folder renamed *jtypes*. Now run *compile.fcc* (which may need to be modified to compile Fortran correctly on the machine you are using) and this turns *makePIs.fcc.f90* into an executable in the user bin called *makePIs.fcc.x*. This program uses *allph00f.in* and *jtypes* as it runs so these must be in the directory that you are creating your new PI file.

The *pi* file labels structures in a unique order based on the size of the unit cell. This labeling system was developed at the National Renewable Energy Laboratory and is currently utilized by a large portion of the cluster expansion community in the U.S. and Germany. The 1st two listed are the 2-atom primitive unit cells. 3-5 are the 3-atom unit cells. 6-17 are the 4-body. 18-31 for 5-atom, 32-81 for 6-atom, 82-133 for 7-atom, 134-375 for 8-atom, 376-627 for 9-atom, and 628-1361 for 10-atom. Found simply by head-tailing the PI file. However, the numbers just given are for a particular run on the 12 atom *PI* file (the one usually used by the group) and can change by a small amount for a different *PI* file even with 12 atoms so that these numbers should be checked whenever a new *PI* file is encountered.

First *cx.in* must be created. To do this, the following must be in the working directory:

```
        fccbin(or bccbin)    str    Atomic.numbers    fort.11    jtypes    LIST    q
```

Inside the *fccbin* folder *predict.pl* is needed that has been checked for correct path names. Specifically, row 37 tells predict where the structure folder is kept. Rows 68 and 69 reference the appropriate phase3 code that may (should) reside in the user bin.

The *str* folder holds the structure data files that will have been created earlier during VASP runs using *pi2str* to create and *makeposcar.pl*, and these file names must be listed in the *LIST* file. The names here must match the *LIST* file names. More on naming these appropriately when *LIST* is discussed

The *Atomic.numbers* file is the easiest file. It is just two rows with the atomic numbers of the cluster expansion alloy. It could easily become confusing as to which atom type should come first. It is the opinion of this author that the lowest atomic number should always come first. For example, for MgO-ZnO, oxygen does not appear in the cluster expansion programming anywhere as it is the spectator atom. It was 'handled' in VASP where the actual energies were calculated, but since it appears in the same place next to every Mg and Zn atom, it does not affect the *geometry* of the problem and therefore does not enter the cluster expansion. So the Atomic.numbers file is just "12 /n 30".

The fort.11 file must be checked. Here is an example:

```
6
J0     J1      3B2-1    4B3-9    4B4-17   5B2-1
-162.1500871  119.5140611 -125.8186719   66.1328898  -33.4798576    6.2615179
6
113.4096117    3.4485786  -70.2915935   34.0391126   -2.1828480   50.0346678
```

The *fort.11* file is extremely important as it contains the fitted coefficients for the MBITs that are created by hand or that the GA selects, and so is essential in calculating the values for every structure in the PI file (i.e. the predictions of the cluster expansion).

Another example:

```
13
 J0     3B3-2    3B3-3    3B3-4    3B4-3    3B5-2    3B5-8    4B3-2    4B3-7    4B4-1
 4B4-6    4B4-10   4B4-17
  2.9154044    0.6042842   -0.5850637   -0.0229367   -0.2176076   -1.8503364
  0.5159651   -0.1199731   -0.3175503   -0.6114110   -0.3214129    1.0985961
  1.8913780
5
  -0.4643694    0.4292195   -0.5600794   -0.5525916    0.5362636
```

None of these numbers is really important at this stage other than they are holding places (this time around). So *predict.pl* is used to predict energies, but during its first use, it is just used as a tool to create the *cx.in* file.

So now run *predict.pl,* but be sure to add the number of structures to the beginning of the *junk* file as the *cx.in* is created as *junk*, and *junk* is missing this key feature.

Now the working directory should look like this:

```
Atomic.numbers  fccbin  fort.11  fort.3  fort.48  LIST  str
cx.in    enpre   fort.1  fort.2   fort.4  jtypes   q     strucot    junk\
```

The *cx.in* should appear as below (well, actually the stuff below the Sk structure stuff is added by hand, usually copying from elsewhere as a template).

```
45
61
'J0      ' 'J1      ' '3B1-1   ' '3B2-1   ' '3B3-1   '
'3B3-2   ' '3B3-3   ' '3B3-4   ' '3B3-5   ' '3B4-1   '
'3B4-2   ' '3B4-3   ' '3B4-4   ' '3B4-5   ' '3B5-1   '
'3B5-2   ' '3B5-3   ' '3B5-4   ' '3B5-5   ' '3B5-6   '
'3B5-7   ' '3B5-8   ' '4B1-1   ' '4B2-1   ' '4B2-2   '
'4B3-1   ' '4B3-2   ' '4B3-3   ' '4B3-4   ' '4B3-5   '
'4B3-6   ' '4B3-7   ' '4B3-8   ' '4B3-9   ' '4B3-10  '
'4B3-11  ' '4B3-12  ' '4B3-13  ' '4B4-1   ' '4B4-2   '
'4B4-3   ' '4B4-4   ' '4B4-5   ' '4B4-6   ' '4B4-7   '
'4B4-8   ' '4B4-9   ' '4B4-10  ' '4B4-11  ' '4B4-12  '
'4B4-13  ' '4B4-14  ' '4B4-15  ' '4B4-16  ' '4B4-17  '
'4B4-18  ' '4B4-19  ' 'M4      ' '5B2-1   ' '6B2-1   '
'J6      '
'mgo'  1
     1    -1    -1    -1    -1    -1    -1    -1    -1    -1
    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1
    -1    -1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1    -1     1
     1
   0    0    0    1 -1.000000000000  0.000000000000
. . .
'struc30zn'  5
    15     9    -3     1     1     5     5     5     9    -3
     1     5     5    -3     3    -1     3     1     5     3
     3     1    -9    -1    -5     3    -5    -1    -1     3
    -1     3     3     7     3     7     3    -1    -5    -1
     3    -5    -1    -1     3     3     3    -1     3     3
     3    -1    -1     7     3     3    -9    -9    -7    -9
     3
   0    0    0    1  0.600000023842  0.000000000000
   0   -2    4    5 -0.400000005960  0.000000000000
   0   -4   -2    5 -0.400000005960  0.000000000000
   0    4    2    5 -0.400000005960  0.000000000000
   0    2   -4    5 -0.400000005960  0.000000000000
'zno'  1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
    1     1     1     1     1     1     1     1     1     1
    1
  0   0   0   1  1.000000000000  0.000000000000
=========t, lambda, pairs, MB settings==================
1 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 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 0 . . .
0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0
1 2 1
13 14 1
1
1 22 1==============Input energies (from LDA, etc.)==========
1     0.0   # TiN
1   -180.4   # L1_1
1   -255.4   # pi3
1   -142.3   # pi3n
. . .
```
(so there is one for every Sk structure above, every 'input' structure you calculated ab initio)
```
1   -44.0   # pi371n
==================leave-many-out-cross-validation==============
14 5
 14  17  21  23  27
```
  . . . (as many as you like, created by *gensets.pl*)
```
  1   4  13  22  31
```

The *cx.in* file should be tested and then used to check the first principles calculations for any transcription errors. Try out the new cx.in file to see if it works by running *cefit.fcc.vb.linux.x < cx.in* where cefit is in the user bin, or a local bin such as *fccbin* so that *./fccbin/cefit.fcc.vb.linux.x < cx.in* is the appropriate command. Here is what should be expected in the working directory afterward:

```
Atomic.numbers   cx.in~   fccbin   fort.1   fort.2   fort.4   JofK   jtypes
LIST   q   strucot   cx.in   enpre   fit5out   fort.11   fort.3   fort.48
jofk.dat   junk   pairs.dat   str
```

Now the genetic algortithm may be run to identify the best choice of multi-body clusters to be used in the cluster expansion model. The cefit comes up with low errors for each structure (maybe less than 10%), but that was using a certain set of MB's that were predetermined with the string of 1's and 0's in the *cx.in* file. Several different MBIT combinations should be tried in order to satisfy oneself that all is going well, and to appreciate the GA since this process of finding the best combination of MBITs used to be done by entirely by hand!

Next the *cx.in* file must be updated for use by the GA so that cefit input section is replaced with the following (or the GA won't be able test its own inputs and thus will fail):

```
. . .
=========t, lambda, pairs, MB settings==================
?mblist?
?lambda?
?t?
1
?pairs?
==============Input energies (from LDA, etc.)=========
```

. . .

Now*./fccbin/findmbs-ga.fcc.pl* can be run, which requires no explicit input, but has several update items of its own. Sections of this are commented for future users, but in case this commented version is lost, here are the most important features:

Lines 43-53, describing the range of parameters to explore with the GA:

```
my $min_lambda = 0.   ;
my $max_lambda = 15.   ;
my $step_lambda = 5.0 ; # For rough sweep try 0 15 5, final sweep 0 15 1.

my $min_t = 0. ;
my $max_t = 40. ;
my $step_t = 10.0 ; # For rough sweep try 0 40 10, final sweep 0 40 2.

my $min_pairs = 1 ;
my $max_pairs = 30 ;
my $step_pairs = 10; # For rough sweep try 1 30 10, final sweep 1 30 2.

my $MBs = "J0.J1" ;
my $pred = "pred-set1" ; #Change this folder-tag for each new run to organize.
my $infile = "cx.in";
```

Line 63 may be changed (repeatedly) for several trial runs.

Lines 88-106, describe other, more complicated working parameters:

```
my $n_genes = 61; # How many MB's did you use?  2nd row cx.in file.
my $n_per_set = 4; # Brian Kolb says 5-10% of structures used.  Check LIST
file.
my $genes_appear = 3; # Aim for around 50 people in your initial population.
    # Calc as n_genes / n_per_set * genes_appear.
my $num_generations = 500; # 500 for rough estimate ~1 hour.  5000 for rough
estimate
    # all night.  5000 final run could take a week!
my $av_mutations_per_gene = 2 ; # 1 or 2 for 60, 3 if you have

my $cross_validation_weight = 1 ;
my $fit_quality_weight = 0 ; # Hart usually zero.
my $fitness_exponent = 2;
my $epsilon = 0.001 ;

my $survival_rate = 0.2;

my $max_locked_in_generations = 50 ;

# stop when the below sigma_pred has been reached; -1.: never stop ...
my $target_sigma_pred = -1. ;
```

Lines 119-133, MBITs to be used must match those used from the pi file:

```
@MB_names = (  'J0     ', 'J1     ', '3B1-1  ', '3B2-1  ', '3B3-1  ',
          '3B3-2  ', '3B3-3  ', '3B3-4  ', '3B3-5  ', '3B4-1  ',
          '3B4-2  ', '3B4-3  ', '3B4-4  ', '3B4-5  ', '3B5-1  ',
          '3B5-2  ', '3B5-3  ', '3B5-4  ', '3B5-5  ', '3B5-6  ',
          '3B5-7  ', '3B5-8  ', '4B1-1  ', '4B2-1  ', '4B2-2  ',
          '4B3-1  ', '4B3-2  ', '4B3-3  ', '4B3-4  ', '4B3-5  ',
          '4B3-6  ', '4B3-7  ', '4B3-8  ', '4B3-9  ', '4B3-10 ',
          '4B3-11 ', '4B3-12 ', '4B3-13 ', '4B4-1  ', '4B4-2  ',
          '4B4-3   ', '4B4-4   ', '4B4-5   ', '4B4-6   ', '4B4-7   ',
          '4B4-8   ', '4B4-9   ', '4B4-10  ', '4B4-11  ', '4B4-12  ',
          '4B4-13  ', '4B4-14  ', '4B4-15  ', '4B4-16  ', '4B4-17  ',
          '4B4-18  ', '4B4-19  ', 'M4      ', '5B2-1   ', '6B2-1   ',
          'J6      '
          ); # This can be created by the jtypes file you are using.
```

Lines 136-142, which MBITs are always turned on during the GA search:

```
my @master_list = ( 1,1,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,
                    0) ;
# List the MBIT's that are always to be used. Must match number of multibody
# interaction types above.  Usually 1,1,zeros...
```

Lines 750-754, be sure home and path are correct:

```
elsif (/lupin/)
  {
    $HOME = "/lupin/rml54/test_minusstr18zn" ;
    $CEFIT = "$HOME/fccbin/cefit.fcc.vb.linux.x" ;
  }
```

Now all that is done is to run the GA with *./fccbin/findmbs-ga.fcc.pl*, but this should be *nohup*'ed as this as it can require a lot of time and the output may want to be examined later. The *nohup* run saves the screen output to the *nohup.out* file, which will be needed to obtain the actual MBITs selected and for use in the cluster expansion. The *nohup.out* file can be checked as it is created by running *tail –f nohup.out*.

With the selected set of MBITs to use, all the structures in the *PI* file now need to be predicted. Here is an example of the solution a GA run gave:

1 1 0 0 0 0 0 0 0 0 0 1 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0

Place this result into *cx.in* and run *cefit.x* just to see the result and check it. But more important, the GA has given the *fort.11* which contains all the information of the CX. So now run *gss.fcc.dat* to create *PIs2thru12.3B5.4B4.5B2.6B2.gss.data*, which needs to be renamed *gss.data*.

Now run *gsl.fcc.pl*. Lines 37 and 38 need to point to the correct PI file:

```
$figure_dir = "/hagrid/hart/MBCE/v0.94/lattice/fcc-based/figures" ;
$infile = "PIs2thru12.3B5.4B4.5B2.6B2" ;
```

Also make sure the user home directory is correctly listed at the very end of the file.

Now the working directory looks like this:

```
Atomic.numbers  enpre   fort.1  fort.48     jtypes
PIs2thru12.3B5.4B4.5B2.6B2.gss.data
cx.in       fccbin   fort.11  gss.data   junk     q    cx.in~    findmbs  fort.2    LIST
str
cx.in.by_hand  findmbs1  fort.3   JofK        nohup.out  strucot   cx.in.GA     findmbs2
fort.4   jofk.dat    pairs.dat
```

Next use the program *findgsl.pl* that sorts the *gss.data* file into *temp.data*. There are modified versions of this floating around, one that removes the colons from the output for use of data on a PC is particularly useful.

This marks the end of the usual path as much scripting that was developed originally creates plots in UNIX by way of *gnuplot*. This user chose to handle the data in *MatLab* on a Macintosh laptop for personal ease. These MatLab scripts are included in this appendix and have some limited documentation though they are fairly straightforward codes. Unfortunately, the previously discussed codes are proprietary and only the user modifiable sections could be delineated.

A.4. MatLab Codes for Data Manipulation

CODE 1

```
%{
Leone, 2006
This code allows for examination of the final
cluster expansion data set, and reduced sets
of structures as well.
%}
close all; clear all; clc;format short g; delta = .00001;
set(0,'defaultaxesfontsize',40);
set(0,'defaulttextfontsize',40);
set(0,'defaultlinelinewidth',1);
indata=fopen('temp.data','r');
M = 375;
A=fscanf(indata,'%g : %g : %g \n');
A=transpose(reshape(A,3,size(A,1)/3));
A(end+1:end+2,:) = [0,7.752700,.5;1,4.987400,-.5];
A(:,1) = 1 - A(:,1);
A=sortrows(A,3);
if mod(size(A,1),2) < delta, %% EVEN
    D=A((size(A,1)/2-M):(size(A,1)/2+M+1),:);
else,  %% ODD
    D=A((floor(size(A,1)/2)-M):ceil((size(A,1)/2)+M),:);
end;

% Remove redundancies on large set.
B = [[A],[abs(A(:,3))]];
B = sortrows(B,4);
B = B(:,1:3);
last = size(B,1);
for i_B = last:-1:2,
    if abs(B(i_B,1)-.5) < delta,
        if abs(abs(B(i_B,3))-abs(B(i_B-1,3))) < delta,
            B(i_B,:) = [];
        end;
    end;
end;
disp(['Large Size = ',num2str(size(B,1))]);

% Reduce large set.
E = sortrows(A,-3);
last = size(E,1);
for i_E = last:-1:2,
    if E(i_E,3) < 0,
        E(i_E,:) = [];
    end;
```

```
end;
disp(['Reduced Large Size = ',num2str(size(E,1))]);

% Remove redundancies on small set.
C = [[D],[abs(D(:,3))]];
C = sortrows(C,4);
C = C(:,1:3);
last = size(C,1);
for i_B = last:-1:2,
    if abs(C(i_B,1)-.5) < delta,
        if abs(abs(C(i_B,3))-abs(C(i_B-1,3))) < delta,
            C(i_B,:) = [];
        end;
    end;
end;
disp(['Small Size = ',num2str(size(C,1))]);

% Reduce small set.
F = sortrows(D,-3);
last = size(F,1);
for i_F = last:-1:2,
    if F(i_F,3) < 0,
        F(i_F,:) = [];
    end;
end;
disp(['Reduced Small Size = ',num2str(size(F,1))]);
fclose all;
```

CODE 2

```
%{
Leone, 2006
This code gives several nice plots of the
input data along with experimentally
determined values.
%}

close all; clear all; clc; format short g;

% load first_data.data;
load nonadj_bg.txt;
nonadj_bg(:,1) = 1 - nonadj_bg(:,1);

% load first_data_offset.data;
load fit5_x_first_pred.data;
```

```
% load experimental_data.data;
load exp_zn.txt;
load exp_mg.txt;

set(0,'defaultaxesfontsize',30);
set(0,'defaulttextfontsize',30);

M = 35;

% Defense
figure(1);
plot(nonadj_bg(:,1),nonadj_bg(:,2),'ks',...
    'markersize',M-15,'MarkerFaceColor','y');
hold on;
title('LDA and Experimental Band Gaps');
xlabel('ZnO Concentration');
ylabel('E_{g} (eV)');
plot(1,.9038,'kh','markersize',M-5,'MarkerFaceColor','y');
plot(exp(2:end,1),exp(2:end,2),'ko',...
    'markersize',M-15,'MarkerFaceColor','g');
plot(exp(1,1),exp(1,2),'kh',...
    'markersize',M-5,'MarkerFaceColor','g');
legend('LDA Rocksalt','LDA Wurtzite','Exp Rocksalt','Exp
Wurtzite',1);
axis([0,1,0,8]);

figure(2);
plot(fit5_x_first_pred(:,1),fit5_x_first_pred(:,2),'ks',...
    'markersize',M-15,'MarkerFaceColor','y');
hold on;
title('Adjusted LDA Band Gaps');
xlabel('ZnO concentration');
ylabel('E_{g} (eV)');
plot(exp(2:end,1),exp(2:end,2),'ko',...
    'markersize',M-15,'MarkerFaceColor','g');
legend('LDA Rocksalt','Exp Rocksalt',4);
axis([0,1,0,8]);


% Paper

set(0,'defaultaxesfontsize',30);
set(0,'defaulttextfontsize',25);

M = 20;
grayset = .3;

figure(1);
```

```
plot(nonadj_bg(:,1),nonadj_bg(:,2),'ks',...
    'markersize',M);
hold on;
title('LDA Calculated Band Gaps','fontsize',35); % Plot all
my RS data
xlabel('ZnO Concentration');
ylabel('E_{g} (eV)');
plot(1,.9038,'kh','markersize',M); % Plot my single LDA WZ
ZnO
legend('LDA Rocksalt','LDA Wurtzite',1);
axis([0,1,0,8]);

figure(2);
plot(nonadj_bg(:,1),nonadj_bg(:,2),'ks',...
    'markersize',M);
hold on;
title({'LDA Calculated and',...
    'Experimentally Measured Band Gaps'},...
    'fontsize',35); % Plot all my RS data
xlabel('ZnO Concentration');
ylabel('E_{g} (eV)');
plot(1,.9038,'kh','markersize',M); % Plot my single LDA WZ
ZnO
plot(exp_mg(:,1),exp_mg(:,2),'ks',... % Experimental RS
measurements

'markersize',M,'MarkerFaceColor',[[grayset],[grayset],[gray
set]]);
plot(exp_zn(:,1),exp_zn(:,2),'kh',... % Experimental WZ
measurements

'markersize',M,'MarkerFaceColor',[[grayset],[grayset],[gray
set]]);
legend('LDA Rocksalt','LDA Wurtzite','Exp Rocksalt','Exp
Wurtzite',1);
axis([0,1,0,8]);

figure(3);
plot(fit5_x_first_pred(:,1),fit5_x_first_pred(:,2),'ks',...
    'markersize',M);
hold on;
title({'Adjusted LDA Calculated and',...
    'Experimentally Measured Rocksalt Band Gaps'},...
    'fontsize',35);
xlabel('ZnO concentration');
ylabel('E_{g} (eV)');
plot(exp_mg(:,1),exp_mg(:,2),'ks',... % Experimental RS
measurements
'markersize',M,'MarkerFaceColor',[[grayset],[grayset],[gray
set]]);
legend('LDA Rocksalt','Exp Rocksalt',4);
axis([0,1,0,8]);
```

CODE 3

```
%{
Leone, 2006
This code produces plots comparing the input
data and the predicted values for those
structures.
%}

close all; clear all; clc; format short g;

load fit5_x_first_pred.data;
load fit5_x_first_pred_add_ons.data;

%PAPER
openmark = 30;
lnwidth = 1;

set(0,'defaultaxesfontsize',30);
set(0,'defaulttextfontsize',30);

figure(1);
plot(fit5_x_first_pred(:,1),fit5_x_first_pred(:,2),'ks',...
    fit5_x_first_pred(:,1),fit5_x_first_pred(:,3),'kx',...
    'markersize',openmark,'linewidth',lnwidth);
hold on;
title({'Cluster Expansions Model Predictions vs.',...
    'LDA Calculations Input to Create Model'});
xlabel('ZnO Concentration');
ylabel('E_{g} (eV)');
legend(' LDA',' Model',1);
axis([0,1,3,8]);
hold off;

figure(2);
plot(fit5_x_first_pred_add_ons(:,1),fit5_x_first_pred_add_o
ns(:,2),'ko',...

fit5_x_first_pred_add_ons(:,1),fit5_x_first_pred_add_ons(:,
3),'k*',...
```

```
    'markersize',openmark,'linewidth',lnwidth);
hold on;
title({'Cluster Expansions Model Predictions',...
    'vs. LDA Calculations of Additional Structures'});
xlabel('ZnO Concentration');
ylabel('E_{g} (eV)');
legend(' LDA',' Model',1);
axis([0,1,3,8]);
hold off;


%DEFENSE
openmark = 40;
lnwidth = 3;

set(0,'defaultaxesfontsize',40);
set(0,'defaulttextfontsize',40);

figure(3);
plot(fit5_x_first_pred(:,1),fit5_x_first_pred(:,2),'ks',...
    fit5_x_first_pred(:,1),fit5_x_first_pred(:,3),'bx',...

'markersize',openmark,'markerfacecolor','y','linewidth',lnw
idth);
hold on;
title({'Cluster Expansions Model Predictions vs.',...
    'LDA Calculations Input to Create Model'});
xlabel('ZnO Concentration');
ylabel('E_{g} (eV)');
legend(' LDA',' Model',1);
axis([0,1,3,8]);
hold off;

figure(4);
plot(fit5_x_first_pred_add_ons(:,1),fit5_x_first_pred_add_o
ns(:,2),'ko',...

fit5_x_first_pred_add_ons(:,1),fit5_x_first_pred_add_ons(:,
3),'r*',...

'markersize',openmark,'markerfacecolor','y','linewidth',lnw
idth);
hold on;
title({'Cluster Expansions Model Predictions',...
    'vs. LDA Calculations of Additional Structures'});
xlabel('ZnO Concentration');
ylabel('E_{g} (eV)');
legend(' LDA',' Model',1);
```

```
axis([0,1,3,8]);
hold off;
```

CODE 4

```
%{
Leone, 2006
This code generates plots of all predicted
structures in the PI file along with the
predicted random alloy values.  It also
gives the predictions on 8 atom or less
primitive cells concurrently with the
experimentally determined values.
%}

close all; clear all; clc;format short g; delta = .00001; M
= 375;
indata=fopen('temp.data','r'); % Get gss results.

A=fscanf(indata,'%g : %g : %g \n');
A=transpose(reshape(A,3,size(A,1)/3));

%!!!!!! Add by hand from fit5out!!!!!!!!!
A(end+1:end+2,:) = [0,7.5666,.5;1,5.0521,-.5];

A(:,1) = 1 - A(:,1);
A=sortrows(A,3);
if mod(size(A,1),2) < delta %% EVEN
    D=A((size(A,1)/2-M):(size(A,1)/2+M),:);
else  %% ODD
    D=A((floor(size(A,1)/2)-M):ceil((size(A,1)/2)+M),:);
end

% return % Turn this on just to get A and D!

load('J_vals.data');
f_RA=inline('ao+a1*(2*x-1)+a2*(2*x-1).^2+a3*(2*x-
1).^3+a4*(2*x-1).^4+a5*(2*x-1).^5+a6*(2*x-1).^6',...
    'x','ao','a1','a2','a3','a4','a5','a6');
x=linspace(0,1,100); i_MBIT = 1;
ao = J_vals(i_MBIT,2);
i_MBIT = i_MBIT + 1;
a1 = J_vals(i_MBIT,2); i_MBIT = i_MBIT + 1;
a2 = 0;
while J_vals(i_MBIT,1) < 3,
    a2 = a2 + J_vals(i_MBIT,2); i_MBIT = i_MBIT + 1;
```

```
end;
a3 = 0;
while J_vals(i_MBIT,1) < 4,
    a3 = a3 + J_vals(i_MBIT,2); i_MBIT = i_MBIT + 1;
end;
a4 = 0;
while J_vals(i_MBIT,1) < 5,
    a4 = a4 + J_vals(i_MBIT,2); i_MBIT = i_MBIT + 1;
end
a5 = 0;
while J_vals(i_MBIT,1) < 6,
    a5 = a5 + J_vals(i_MBIT,2); i_MBIT = i_MBIT + 1;
end
a6 = 0;
while J_vals(i_MBIT,1) < 7,
    a6 = a6 + J_vals(i_MBIT,2); i_MBIT = i_MBIT + 1;
end
Eg_RA = f_RA(x,ao,a1,a2,a3,a4,a5,a6);


% Defense.
set(0,'defaultaxesfontsize',40);
set(0,'defaulttextfontsize',40);
set(0,'defaultlinelinewidth',1);
MM = 15;

figure(1);
plot(1-A(:,1),A(:,2),'ko','markersize',MM-5,...
    'markerfacecolor','y');
hold on;
plot(x,Eg_RA,'b-','linewidth',4);
legend([{'Specific Structures','Random
Alloy'}],1,'fontsize',25);
title({'Band Gap Predictions for the 16,089',...
    'Primitive Cells of 13 or Less Non-Spectator
Atoms'},...
    'fontsize',30);
xlabel('ZnO Concentration');
ylabel('E_{g} (eV)');
axis([0,1,2,10]);
hold off;

figure(2);
plot(1-A(:,1),A(:,2),'ko','markersize',MM-5,...
    'markerfacecolor','y');
hold on;
plot(x,Eg_RA,'b-','linewidth',4);
legend([{'Specific Structures','Random
Alloy'}],3,'fontsize',25);
title({'Band Gap Predictions for the 9,008',...
    'Primitive Cells of 13 or Less Non-Spectator Atoms',...
    'within ZnO Concentration Range Forming Rocksalt'},...
    'fontsize',30);
xlabel('Restricted ZnO Concentration');
ylabel('E_{g} (eV)');
axis([0,.5,2,10]);
hold off;

figure(3);
plot(1-D(:,1),D(:,2),'ko','markersize',MM,...
    'markerfacecolor','y');
hold on;
plot(x,Eg_RA,'b-','linewidth',MM-8);
load exp.txt;
plot(1-exp(:,1),exp(:,2),...
    'gs','markersize',MM+10,'markeredgecolor','k',...
    'markerfacecolor','g');
legend([{'Model','Random
Alloy','Experimental'}],3,'fontsize',25);
title({'Band Gap Predictions for the 376',...
    'Primitive Cells of 8 or Less Non-Spectator Atoms',...
    'within ZnO Concentration Range Forming Rocksalt'},...
    'fontsize',30);
xlabel('Restricted ZnO Concentration');
ylabel('E_{g} (eV)');
axis([0,.5,2,10]);
hold off;



% Paper.
set(0,'defaultaxesfontsize',30);
set(0,'defaulttextfontsize',25);
set(0,'defaultlinelinewidth',1);
MM = 15;

figure(4);
plot(1-A(:,1),A(:,2),'ko','markersize',MM-5);
hold on;
plot(x,Eg_RA,'k-','linewidth',4);
legend([{'Specific Structures','Random
Alloy'}],1,'fontsize',25);
title({'Band Gap Predictions for the 16,089',...
    'Primitive Cells of 13 or Less Non-Spectator
Atoms'},...
    'fontsize',30);
```

```
xlabel('ZnO Concentration');
ylabel('E_{g} (eV)');
axis([0,1,2,10]);
hold off;

figure(5);
plot(1-A(:,1),A(:,2),'ko','markersize',MM-5);
hold on;
plot(x,Eg_RA,'k-','linewidth',4);
legend([{'Specific Structures','Random
Alloy'}],3,'fontsize',25);
title({'Band Gap Predictions for the 9,008',...
'Primitive Cells of 13 or Less Non-Spectator Atoms',...
    'within ZnO Concentration Range Forming Rocksalt'},...
    'fontsize',30);
xlabel('Restricted ZnO Concentration');
ylabel('E_{g} (eV)');
axis([0,.5,2,10]);
hold off;
M_color = .2;
figure(6);
plot(1-D(:,1),D(:,2),'ko','markersize',MM-7);%,...
%     'markeredgecolor',[M_color,M_color,M_color]);
hold on;
plot(x,Eg_RA,'k-','linewidth',MM-
13,'color',[M_color,M_color,M_color]);
load exp_mg.txt;
plot(exp_mg(:,1),exp_mg(:,2),...
    'ks','markersize',MM+2,...
    'markeredgecolor','k','markerfacecolor',[.5,.5,.5]);
legend([{'Model','Random
Alloy','Experimental'}],3,'fontsize',25);
title({'Band Gap Predictions for the 376',...
    'Primitive Cells of 8 or Less Non-Spectator Atoms',...
    'within ZnO Concentration Range Forming Rocksalt'},...
    'fontsize',35);
xlabel('Restricted ZnO Concentration');
ylabel('E_{g} (eV)');
axis([0,.5,2,10]);
hold off;

fclose all;
```

CODE 5

```
%{
Leone, 2006
This program creates nice pseudo 3-D images
of crystal structures using the lattice vectors
and atomic basis vectors from the POSCAR file.
Tweaking of the input section is a must.
%}

close all; clear all; clc; format short g; delta = 1e-3;

%%%%%%%%%%% INPUTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% x-axis:L-R, y-axis:I-O, z-axis;U-D
% Adjust in steps, don't use y-axis.
% 100=0,0,0; 111=0,0,45; 110=90,0,90; 211=0,0,45
% 311=0,0,45; 320=0,0,0; 210=0,0,90; 310=0,0,90
% 410=0,0,90; 331=135,90,90; 531=55,0,45
% Str 80: x=235,y=0,z=48, shut off view and adjust axis at
end.
%
theta_x = 55;
theta_y = 0;
theta_z = 45;
edge_length = 10;
axisedge = 1.5;
PE = axisedge;
MM = 40; % (axisedge,MM) (1.25,65) (1.5,40-55) (2.0,25-35)

%%%%%%%%%%% INSERT STRUCTURE %%%%%%%%%%%%%%%%%%%%%%%%%
bg = 6.01;
stacking = ['A_{3}B_{4} Stacking (531)'];
% % 132, 531, a3b4
LV=[
  1.000000,   0.500000,   0.500000
 -0.500000,   1.000000,   0.500000
 -0.500000,  -0.500000,   1.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   0.000000,   0.000000,   1.000000
  30   0.500000,   0.500000,   1.000000
  12   0.000000,   0.500000,   0.500000
  12   0.000000,   0.500000,   1.500000
  12  -0.500000,   0.500000,   1.000000
  12   0.000000,   1.000000,   1.000000
];
%%%%%%%%%%% END POSCAR %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%% STUFF %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ABV = sortrows(ABV,1);
if mod(edge_length,2) > delta,
    disp('''edge_length'' not an even number.  Redo.');
    return;
end;


%%%%%%%%%%% Get N's %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N_atoms = size(ABV,1);
N_mg = 1;
while ABV(N_mg,1) < 30,
    N_mg = N_mg + 1;
end;
N_mg = N_mg - 1;
N_zn = N_atoms - N_mg;
ABV = ABV(:,2:4);


%%%%%%%%%%% Rotate %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
theta_x = theta_x*pi/180;
theta_y = theta_y*pi/180;
theta_z = theta_z*pi/180;
R_x = [1,0,0;
    0,cos(theta_x),-sin(theta_x);
    0,sin(theta_x),cos(theta_x)]; % X-axis.
R_y = [cos(theta_y),0,sin(theta_y);
    0,1,0;
    -sin(theta_y),0,cos(theta_y)]; % Y-axis.
R_z = [cos(theta_z),-sin(theta_z),0;
    sin(theta_z),cos(theta_z),0;
    0,0,1]; % Z-axis.
LV = transpose(R_x*R_y*R_z*LV');
ABV = transpose(R_x*R_y*R_z*ABV');


%%%%%%%%%%% Make MG atoms %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MG = zeros(N_mg*(edge_length+1)^3,3);
ind = 1;
for i_x = -edge_length/2:edge_length/2,
    for i_y = -edge_length/2:edge_length/2,
        for i_z = -edge_length/2:edge_length/2,
            for i_mg = 1:N_mg
                MG(ind,:) = i_x*LV(1,:) + i_y*LV(2,:) + ...
                    i_z*LV(3,:) + ABV(i_mg,:);
                if (abs(MG(ind,1))>PE ||...
                        abs(MG(ind,2))>PE ||...
                        abs(MG(ind,3))>PE),
                     MG(ind,:) = ABV(i_mg,:);
                 end;
                 ind = ind + 1;
             end;
         end;
     end;
end;
    % Remove duplicates.
MG = sortrows(MG,[1,2,3]);
i_mg = size(MG,1);
while i_mg > 1,
    if sum(abs(MG(i_mg,:) - MG(i_mg-1,:))) < delta,
        MG(i_mg,:) = [];
    end;
    i_mg = i_mg - 1;
end;
    % Remove out-of-bounds.
i_mg = size(MG,1);
while i_mg > 0,
    if abs(MG(i_mg,1)) > axisedge + delta ||...
            abs(MG(i_mg,2)) > axisedge + delta ||...
            abs(MG(i_mg,3)) > axisedge + delta,
        MG(i_mg,:) = [];
    end;
    i_mg = i_mg - 1;
end;


%%%%%%%%%%% Make Zn atoms %%%%%%%%%%%%%%%%%%%%%%%%%%%%
ZN = zeros(N_zn*(edge_length+1)^3,3);
ind = 1;
for i_x = -edge_length/2:edge_length/2,
    for i_y = -edge_length/2:edge_length/2,
        for i_z = -edge_length/2:edge_length/2,
            for i_zn = N_mg+1:N_atoms,
                ZN(ind,:) = i_x*LV(1,:) + i_y*LV(2,:) + ...
                    i_z*LV(3,:) + ABV(i_zn,:);
                if (abs(ZN(ind,1))>PE ||...
                       abs(ZN(ind,2))>PE ||...
                       abs(ZN(ind,3))>PE),
                    ZN(ind,:) = ABV(i_zn,:);
                end;
                ind = ind + 1;
            end;
        end;
    end;
end;
```

```
    % Remove duplicates.
ZN = sortrows(ZN,[1,3,2]);
i_zn = size(ZN,1);
while i_zn > 1,
    if sum(abs(ZN(i_zn,:) - ZN(i_zn-1,:))) < delta,
        ZN(i_zn,:) = [];
    end;
    i_zn = i_zn - 1;
end;
    % Remove out-of-bounds.
i_zn = size(ZN,1);
while i_zn > 0,
    if abs(ZN(i_zn,1)) > axisedge + delta ||...
            abs(ZN(i_zn,2)) > axisedge + delta ||...
            abs(ZN(i_zn,3)) > axisedge + delta,
        ZN(i_zn,:) = [];
    end;
    i_zn = i_zn - 1;
end;


%%%%%%%%%%%% Draw Structure Defense %%%%%%%%%%%%%%%%%%%%%
set(gcf,'units','normalized','position',[.01,.05,.5,.9]);
set(0,'defaultaxesfontsize',40);
set(0,'defaulttextfontsize',40);
set(0,'defaultlinelinewidth',2);

% Minor Rotate
theta_x = 3*pi/180;
theta_y = 0*pi/180;
theta_z = 0*pi/180;
R_x = [1,0,0;
    0,cos(theta_x),-sin(theta_x);
    0,sin(theta_x),cos(theta_x)]; % X-axis.
R_y = [cos(theta_y),0,sin(theta_y);
    0,1,0;
    -sin(theta_y),0,cos(theta_y)]; % Y-axis.
R_z = [cos(theta_z),-sin(theta_z),0;
    sin(theta_z),cos(theta_z),0;
    0,0,1]; % Z-axis.
MG = transpose(R_x*R_y*R_z*MG');
ZN = transpose(R_x*R_y*R_z*ZN');


MG(:,4) = 1*ones(size(MG,1),1);
ZN(:,4) = -1*ones(size(ZN,1),1);
MGZN = [[MG];[ZN]];
MGZN = sortrows(MGZN,3);
```

```
grayc = .5;
if abs(MGZN(1,4)-1) < delta,
    plot3(MGZN(1,1),MGZN(1,2),...
        MGZN(1,3),'ro','markersize',MM,...
        'markerfacecolor','w','markeredgecolor','k');
else,
    plot3(MGZN(1,1),MGZN(1,2),...
        MGZN(1,3),'bo','markersize',MM,...
        'markerfacecolor',[grayc,grayc,grayc],...
        'markeredgecolor','k');
end;
hold on;
for i_mgzn = 2:size(MGZN,1);
    if abs(MGZN(i_mgzn,4)-1) < delta,
        plot3(MGZN(i_mgzn,1),MGZN(i_mgzn,2),...
            MGZN(i_mgzn,3),'ro','markersize',MM,...
            'markerfacecolor','w','markeredgecolor','k');
    else,
        plot3(MGZN(i_mgzn,1),MGZN(i_mgzn,2),...
            MGZN(i_mgzn,3),'bo','markersize',MM,...
            'markerfacecolor',[grayc,grayc,grayc],...
            'markeredgecolor','k');
    end;
end;
drawnow;
axis([min(MGZN(:,1)),max(MGZN(:,1)),min(MGZN(:,2)),...
    max(MGZN(:,2)),min(MGZN(:,3)),max(MGZN(:,3))]);
% axis([min(MGZN(:,1))-.3,max(MGZN(:,1))+.3,min(MGZN(:,2))-
.3,...
%     max(MGZN(:,2))+.3,min(MGZN(:,3))-
.3,max(MGZN(:,3))+.3]);
grid on;
drawnow;
hold off;
view([0,0]); % Toggle
% axis([-.75,.75,-.75,.75,-.75,.75]); % Toggle
title({[[num2str(bg)],' eV',' Band Gap '],...
    [stacking],[' ']},'fontsize',35);
drawnow;
axis off;
```

CODE 6

```
%{
Leone, 2006
This file contains many of the crystal
structures' POSCAR and other data that
were used in my research for plotting.
By databasing these structures, I can
come back to them later if a better
plot is needed or if the predicted
values should change.  Use this with the
previous viewing program.
%}

%% Crystal structures to be plotted.
%% Contents:
%% 1, 2, 3, 4, 8, 10, 11, 15
%% 16, 17, 19, 21, 22, 30, 32, 37,
%% 42, 45, 58, 63, 67, 70, 72, 73,
%% 75, 80, 86, 90, 107, 108, 109,
%% 115, 123, 125, 132, 142, 149,
%% 156, 190, 193, 208, 287, 289, 290,
%% 303, 322, 585, 1138, 1853, 4688

% % 1, 100, a1b1
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,  -0.500000,   0.000000
  0.000000,   0.000000,  -1.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.000000,  -0.500000
];


% % 2, 111, a1b1
LV=[
  0.50000000   0.50000000   0.00000000
  0.50000000   0.00000000   0.50000000
  1.00000000  -0.50000000  -0.50000000
];
ABV=[
 30   0.000000    0.000000    0.000000
 12   1.000000    0.000000    0.000000
];


% % 3, 110, a1b2
```

```
LV=[
  0.500000,   0.500000,   0.000000
  0.000000,   0.000000,   1.000000
  1.000000,  -0.500000,   0.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.000000,   0.500000
  12   1.000000,   0.000000,   1.000000
];


% % 4, 100, a1b2
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,  -0.500000,   0.000000
  0.500000,   0.000000,  -1.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   1.000000,   0.000000,  -1.000000
  12   0.500000,   0.000000,  -0.500000
];


% % 8, 100, a1b3
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,  -0.500000,   0.000000
  0.000000,   0.000000,  -2.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.000000,  -1.500000
  12   0.000000,   0.000000,  -1.000000
  12   0.500000,   0.000000,  -0.500000
];


% % 10, SS
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,  -0.500000,   1.000000
  0.500000,  -0.500000,  -1.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.000000,   0.500000
  12   0.500000,   0.000000,  -0.500000
  12   0.500000,  -0.500000,   0.000000
];


% % 11, 311, a1b3
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,  -0.500000,   1.000000
  1.000000,  -0.500000,  -0.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.000000,   0.500000
  12   1.000000,   0.000000,   0.000000
  12   1.000000,  -0.500000,   0.500000
];


% % 15, SS
LV=[
  1.000000,   0.000000,   0.000000
  0.000000,   1.000000,   0.000000
  0.000000,   0.000000,   1.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.500000,   0.000000
  12   0.500000,   0.000000,   0.500000
  12   0.000000,   0.500000,   0.500000
];


% % 16, 210, a1b3
LV=[
  1.000000,   0.000000,   0.000000
  0.000000,   1.000000,   0.000000
  0.500000,   0.500000,   1.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.500000,   0.000000
  12   0.500000,   1.000000,   0.500000
  12   1.000000,   0.500000,   0.500000
];


% % 17, 210, a2b2
```

```
LV=[
  1.000000,   0.000000,   0.000000
  0.000000,   1.000000,   0.000000
  0.500000,   0.500000,   1.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.500000,   0.000000
  30   0.500000,   1.000000,   0.500000
  12   1.000000,   0.500000,   0.500000
];


% % 19, 110, a2b3
LV=[
  0.500000,   0.500000,   0.000000
  0.000000,   0.000000,   1.000000
  1.500000,  -1.000000,   0.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   0.500000,   0.000000,   0.500000
  12   1.000000,   0.000000,   1.000000
  12   1.000000,  -0.500000,   0.500000
  12   1.500000,  -0.500000,   1.000000
];


% % 21, 100, a1b4
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,  -0.500000,   0.000000
  0.500000,   0.000000,  -2.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   1.000000,   0.000000,  -2.000000
  12   0.500000,   0.000000,  -1.500000
  12   1.000000,   0.000000,  -1.000000
  12   0.500000,   0.000000,  -0.500000
];


% % 22, 100, a2b3
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,  -0.500000,   0.000000
  0.500000,   0.000000,  -2.500000
```

```
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   1.000000,   0.000000,  -2.000000
  12   0.500000,   0.000000,  -1.500000
  12   1.000000,   0.000000,  -1.000000
  12   0.500000,   0.000000,  -0.500000
];


% % 30, 210, a1b4
LV=[
  1.000000,   0.000000,   0.000000
  0.500000,   1.000000,   0.500000
  0.500000,  -0.500000,   1.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   1.000000,   0.000000,   1.000000
  12   1.500000,   0.500000,   1.000000
  12   0.500000,   0.000000,   0.500000
  12   1.000000,   0.500000,   0.500000
];


% % 32, 110, a1b5
LV=[
  0.500000,   0.500000,   0.000000
  0.000000,   0.000000,   1.000000
  1.500000,  -1.500000,   0.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.000000,   0.500000
  12   0.500000,  -0.500000,   0.000000
  12   1.000000,  -0.500000,   0.500000
  12   1.000000,  -1.000000,   0.000000
  12   1.500000,  -1.000000,   0.500000
];


% % 37, 110, a1b5
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,  -0.500000,   0.000000
  0.000000,   0.000000,  -3.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.000000,  -2.500000
  12   0.000000,   0.000000,  -2.000000
  12   0.500000,   0.000000,  -1.500000
  12   0.000000,   0.000000,  -1.000000
  12   0.500000,   0.000000,  -0.500000
];


% % 42, 211, a1b5
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,  -0.500000,   1.000000
  1.000000,  -1.000000,  -1.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.000000,   0.500000
  12   1.000000,  -0.500000,  -0.500000
  12   1.000000,  -1.000000,   0.000000
  12   0.500000,  -0.500000,   0.000000
  12   1.000000,  -0.500000,   0.500000
];


% % 45, 211, a1b1a1b2
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,  -0.500000,   1.000000
  1.000000,  -1.000000,  -1.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.000000,   0.500000
  12   1.000000,  -0.500000,  -0.500000
  30   1.000000,  -1.000000,   0.000000
  12   0.500000,  -0.500000,   0.000000
  12   1.000000,  -0.500000,   0.500000
];


% % 58, 331, ?
LV=[
  0.500000,   0.500000,   0.000000
  1.000000,  -1.000000,   0.000000
  0.000000,   0.500000,  -1.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   0.500000,   0.500000,  -1.000000
  12   0.500000,   0.000000,  -0.500000
  12   0.500000,  -0.500000,   0.000000
  12   1.000000,   0.000000,  -1.000000
  12   1.000000,  -0.500000,  -0.500000
];


% % 63, 311, a2b4
LV=[
  0.500000,   0.500000,   0.000000
  1.000000,  -0.500000,   0.500000
  1.000000,  -0.500000,  -1.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   1.500000,  -0.500000,  -1.000000
  12   1.000000,   0.000000,  -1.000000
  12   1.000000,  -0.500000,  -0.500000
  12   0.500000,   0.000000,  -0.500000
  12   1.000000,   0.000000,   0.000000
];


% % 67, 320, a1b5
LV=[
  1.000000,   0.000000,   0.000000
  0.000000,   1.000000,   0.000000
  0.000000,   0.500000,   1.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.500000,   0.000000
  12   0.500000,   1.000000,   0.500000
  12   0.000000,   0.500000,   0.500000
  12   0.000000,   1.000000,   1.000000
  12   0.500000,   0.500000,   1.000000
];


% % 70, 320, a1b1a1b3
LV=[
  1.000000,   0.000000,   0.000000
  0.000000,   1.000000,   0.000000
  0.000000,   0.500000,   1.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.500000,   0.000000
  12   0.500000,   1.000000,   0.500000
  30   0.000000,   0.500000,   0.500000
  12   0.000000,   1.000000,   1.000000
  12   0.500000,   0.500000,   1.000000
];


% % 72, 210, a1b5
LV=[
  1.000000,   0.000000,   0.000000
  0.500000,   1.000000,   0.500000
  0.000000,  -1.000000,   1.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,  -0.500000,   1.000000
  12   1.000000,   0.000000,   1.000000
  12   0.000000,  -0.500000,   0.500000
  12   0.500000,   0.000000,   0.500000
  12   1.000000,   0.500000,   0.500000
];


% % 73, 210, a1b1a1b3
LV=[
  1.000000,   0.000000,   0.000000
  0.500000,   1.000000,   0.500000
  0.000000,  -1.000000,   1.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   0.500000,  -0.500000,   1.000000
  12   1.000000,   0.000000,   1.000000
  12   0.000000,  -0.500000,   0.500000
  12   0.500000,   0.000000,   0.500000
  12   1.000000,   0.500000,   0.500000
];


% % 75, 210, a1b2a1b2
LV=[
  1.000000,   0.000000,   0.000000
  0.500000,   1.000000,   0.500000
  0.000000,  -1.000000,   1.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   0.500000,  -0.500000,   1.000000
  30   1.000000,   0.000000,   1.000000
  12   0.000000,  -0.500000,   0.500000
  12   0.500000,   0.000000,   0.500000
  12   1.000000,   0.500000,   0.500000
];


% % 80, 531, a1b1a2b2
LV=[
  1.000000,   0.500000,   0.500000
 -0.500000,   1.000000,   0.500000
 -0.500000,  -1.000000,   0.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   0.000000,  -0.500000,   0.500000
  12   0.500000,   0.000000,   0.500000
  30   0.000000,   0.500000,   0.500000
  12   0.000000,   0.000000,   1.000000
  12  -0.500000,   0.000000,   0.500000
];


% % 86, 110, a1b2a1b3
LV=[
  0.500000,   0.500000,   0.000000
  0.000000,   0.000000,   1.000000
  2.000000,  -1.500000,   0.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.000000,   0.500000
  12   1.000000,   0.000000,   1.000000
  30   1.000000,  -0.500000,   0.500000
  12   1.500000,  -0.500000,   1.000000
  12   1.500000,  -1.000000,   0.500000
  12   2.000000,  -1.000000,   1.000000
];


% % 90, 100, a1b6
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,  -0.500000,   0.000000
  0.500000,   0.000000,  -3.500000
];
```

```
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   1.000000,   0.000000,  -3.000000
  12   0.500000,   0.000000,  -2.500000
  12   1.000000,   0.000000,  -2.000000
  12   0.500000,   0.000000,  -1.500000
  12   1.000000,   0.000000,  -1.000000
  12   0.500000,   0.000000,  -0.500000
];


% % 107, 331, a2b5
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,   0.000000,   1.500000
  1.000000,  -1.000000,  -1.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   0.500000,   0.000000,   0.500000
  12   1.000000,   0.000000,   1.000000
  12   1.000000,   0.000000,   0.000000
  12   1.000000,  -0.500000,   0.500000
  12   1.000000,  -0.500000,  -0.500000
  12   1.500000,  -0.500000,   0.000000
];


% % 108, 331, a1b1a1b4
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,   0.000000,   1.500000
  1.000000,  -1.000000,  -1.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.000000,   0.500000
  30   1.000000,   0.000000,   1.000000
  12   1.000000,   0.000000,   0.000000
  12   1.000000,  -0.500000,   0.500000
  12   1.000000,  -0.500000,  -0.500000
  12   1.500000,  -0.500000,   0.000000
];


% % 109, 331, a3b4
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,   0.000000,   1.500000
  1.000000,  -1.000000,  -1.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   0.500000,   0.000000,   0.500000
  30   1.000000,   0.000000,   1.000000
  12   1.000000,   0.000000,   0.000000
  12   1.000000,  -0.500000,   0.500000
  12   1.000000,  -0.500000,  -0.500000
  12   1.500000,  -0.500000,   0.000000
];


% % 115, 311, a2b5
LV=[
  0.500000,   0.500000,   0.000000
  1.000000,  -0.500000,   0.500000
  0.500000,  -0.500000,  -2.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   1.000000,  -0.500000,  -1.500000
  12   1.500000,  -0.500000,  -1.000000
  12   1.000000,   0.000000,  -1.000000
  12   1.000000,  -0.500000,  -0.500000
  12   0.500000,   0.000000,  -0.500000
  12   1.000000,   0.000000,   0.000000
];


% % 123, 210, a2b5
LV=[
  1.000000,   0.000000,   0.000000
  0.500000,   1.000000,   0.500000
  0.000000,  -0.500000,   1.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   0.500000,   0.000000,   1.500000
  12   1.000000,   0.500000,   1.500000
  12   1.000000,   0.000000,   1.000000
  12   0.500000,   0.500000,   1.000000
  12   0.500000,   0.000000,   0.500000
  12   1.000000,   0.500000,   0.500000
];
```

```
% % 125, 210, a3b4
LV=[
  1.000000,   0.000000,   0.000000
  0.500000,   1.000000,   0.500000
  0.000000,  -0.500000,   1.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   0.500000,   0.000000,   1.500000
  30   1.000000,   0.500000,   1.500000
  12   1.000000,   0.000000,   1.000000
  12   0.500000,   0.500000,   1.000000
  12   0.500000,   0.000000,   0.500000
  12   1.000000,   0.500000,   0.500000
];


% % 132, 531, a3b4
LV=[
  1.000000,   0.500000,   0.500000
 -0.500000,   1.000000,   0.500000
 -0.500000,  -0.500000,   1.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   0.000000,   0.000000,   1.000000
  30   0.500000,   0.500000,   1.000000
  12   0.000000,   0.500000,   0.500000
  12   0.000000,   0.500000,   1.500000
  12  -0.500000,   0.500000,   1.000000
  12   0.000000,   1.000000,   1.000000
];


% % 142, 110,
LV=[
  0.500000,   0.500000,   0.000000
  0.000000,   0.000000,   1.000000
  2.000000,  -2.000000,   0.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.000000,   0.500000
  30   0.500000,  -0.500000,   0.000000
  12   1.000000,  -0.500000,   0.500000
  30   1.000000,  -1.000000,   0.000000
  12   1.500000,  -1.000000,   0.500000
  12   1.500000,  -1.500000,   0.000000
  12   2.000000,  -1.500000,   0.500000
];


% % 149, 100, a1b7
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,  -0.500000,   0.000000
  0.000000,   0.000000,  -4.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.000000,  -3.500000
  12   0.000000,   0.000000,  -3.000000
  12   0.500000,   0.000000,  -2.500000
  12   0.000000,   0.000000,  -2.000000
  12   0.500000,   0.000000,  -1.500000
  12   0.000000,   0.000000,  -1.000000
  12   0.500000,   0.000000,  -0.500000
];


% % 156, 100, a2b2a1b3
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,  -0.500000,   0.000000
  0.000000,   0.000000,  -4.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   0.500000,   0.000000,  -3.500000
  12   0.000000,   0.000000,  -3.000000
  12   0.500000,   0.000000,  -2.500000
  30   0.000000,   0.000000,  -2.000000
  12   0.500000,   0.000000,  -1.500000
  12   0.000000,   0.000000,  -1.000000
  12   0.500000,   0.000000,  -0.500000
];


% % 190, 533, a1b1a1b1a1b1a1b1
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,  -0.500000,   1.000000
  1.500000,  -1.000000,  -1.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.000000,   0.500000
  30   1.500000,  -0.500000,  -1.000000
  12   1.500000,  -1.000000,  -0.500000
  30   1.000000,  -0.500000,  -0.500000
  12   1.500000,  -0.500000,   0.000000
  30   1.000000,   0.000000,   0.000000
  12   1.000000,  -0.500000,   0.500000
];


% % 193, 111, a2b6
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,   0.000000,   0.500000
  3.000000,  -2.500000,  -2.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   3.000000,  -2.000000,  -2.000000
  12   3.000000,  -1.500000,  -1.500000
  12   2.000000,  -1.500000,  -1.500000
  12   2.000000,  -1.000000,  -1.000000
  12   2.000000,  -0.500000,  -0.500000
  12   1.000000,  -0.500000,  -0.500000
  12   1.000000,   0.000000,   0.000000
];


% % 208, 331, a2b6
LV=[
  0.500000,   0.500000,   0.000000
  0.500000,   0.000000,   1.500000
  1.000000,  -1.500000,  -0.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   0.500000,   0.000000,   0.500000
  12   1.000000,   0.000000,   1.000000
  12   0.500000,  -0.500000,   0.000000
  12   1.000000,  -0.500000,   0.500000
  12   1.500000,  -0.500000,   1.000000
  12   1.000000,  -1.000000,   0.000000
  12   1.500000,  -1.000000,   0.500000
];


% % 287, 410, a1b1a1b2a1b2
LV=[
  1.000000,   0.000000,   0.000000
  0.000000,   1.000000,   0.000000
  0.500000,   0.500000,   2.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.500000,   0.000000
  30   0.500000,   1.000000,   0.500000
  12   1.000000,   0.500000,   0.500000
  12   1.000000,   1.000000,   1.000000
  30   0.500000,   0.500000,   1.000000
  12   0.500000,   1.000000,   1.500000
  12   1.000000,   0.500000,   1.500000
];


% % 289, 410, ?
LV=[
  1.000000,   0.000000,   0.000000
  0.000000,   1.000000,   0.000000
  0.500000,   0.500000,   2.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,   0.500000,   0.000000
  30   0.500000,   1.000000,   0.500000
  12   1.000000,   0.500000,   0.500000
  30   1.000000,   1.000000,   1.000000
  12   0.500000,   0.500000,   1.000000
  30   0.500000,   1.000000,   1.500000
  12   1.000000,   0.500000,   1.500000
];


% % 290, SS
LV=[
  1.000000,   0.000000,   0.000000
  0.000000,   1.000000,   1.000000
  0.000000,  -1.000000,   1.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  12   0.500000,  -0.500000,   1.000000
  12   0.000000,   0.000000,   1.000000
  12   0.500000,   0.500000,   1.000000
  12   0.500000,   0.000000,   0.500000
  12   0.000000,   0.500000,   0.500000
  12   0.500000,   0.000000,   1.500000
  12   0.000000,  -0.500000,   0.500000
];


% % 303, 310, a4b4
LV=[
  1.000000,   0.000000,   0.000000
  0.000000,   1.000000,   1.000000
  0.000000,  -0.500000,   1.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   0.500000,   0.000000,   1.500000
  30   0.000000,   0.500000,   1.500000
  30   0.500000,   0.000000,   0.500000
  12   0.000000,   0.500000,   0.500000
  12   0.500000,   0.500000,   2.000000
  12   0.000000,   0.000000,   1.000000
  12   0.500000,   0.500000,   1.000000
];


% % 322, 210, a2b1a1b1a1b2
LV=[
  1.000000,   0.000000,   0.000000
  0.500000,   1.000000,   0.500000
 -0.500000,  -1.000000,   1.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   0.000000,  -0.500000,   1.500000
  12   0.500000,   0.000000,   1.500000
  30   0.500000,  -0.500000,   1.000000
  12   0.000000,   0.000000,   1.000000
  30   0.500000,   0.500000,   1.000000
  12   0.500000,   0.000000,   0.500000
  12   1.000000,   0.500000,   0.500000
];


% % 585, SS
LV=[
  1.00000000   0.00000000   0.00000000
  0.50000000   1.50000000   0.00000000
  0.50000000   0.00000000   1.50000000
];
ABV=[
 30   0.000000    0.000000    0.000000
 30   0.500000    0.500000    0.000000
 12   1.000000    1.000000    0.000000
 30   0.500000    0.000000    0.500000
 30   1.000000    0.500000    0.500000
 12   0.500000    1.000000    0.500000
 12   1.000000    0.000000    1.000000
 12   0.500000    0.500000    1.000000
 12   1.000000    1.000000    1.000000
];


% % 1138,
LV=[
  1.00000000   0.00000000   0.00000000
  0.00000000   1.50000000   0.50000000
  0.00000000  -0.50000000   1.50000000
];
ABV=[
 30   0.000000    0.000000    0.000000
 12   0.500000    0.000000    1.500000
 30   0.000000    0.500000    1.500000
 12   0.500000    1.000000    1.500000
 30   0.000000    0.000000    1.000000
 12   0.500000    0.500000    1.000000
 12   0.000000    1.000000    1.000000
 12   0.500000    0.000000    0.500000
 12   0.000000    0.500000    0.500000
 12   0.500000    1.000000    0.500000
];


% % 1853, 410, a3b2a1b2a1b2
LV=[
  1.000000,   0.000000,   0.000000
  0.000000,   1.500000,   0.500000
  0.500000,  -1.000000,   1.500000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   1.000000,  -0.500000,   1.500000
  30   0.500000,   0.000000,   1.500000
  12   1.000000,   0.500000,   1.500000
  12   0.500000,  -0.500000,   1.000000
  30   1.000000,   0.000000,   1.000000
  12   0.500000,   0.500000,   1.000000
  12   1.000000,   1.000000,   1.000000
  30   0.500000,   0.000000,   0.500000
  12   1.000000,   0.500000,   0.500000
  12   0.500000,   1.000000,   0.500000
];


% % 4688, 410, a4b2a1b2a1b2
LV=[
  1.000000,   0.000000,   0.000000
  0.500000,   1.500000,   0.000000
 -0.500000,  -0.500000,   2.000000
];
ABV=[
  30   0.000000,   0.000000,   0.000000
  30   0.500000,   0.500000,   0.000000
  30   1.000000,   1.000000,   0.000000
  30   0.500000,   0.000000,   0.500000
  12   1.000000,   0.500000,   0.500000
  12   0.500000,   1.000000,   0.500000
  30   0.000000,   0.000000,   1.000000
  12   0.500000,   0.500000,   1.000000
  12   1.000000,   1.000000,   1.000000
  30   0.500000,   0.000000,   1.500000
  12   0.000000,   0.500000,   1.500000
  12   0.500000,   1.000000,   1.500000
];
```

CODE 7

```
%{
Leone, 2006
This code is a last resort for determining
what rotation to use to plot tricky stacking
sequences for tough crystal structures.  It
rotates the image until a good vantage is
determined.  Use this with the previous two
viewing programs.
%}

clc;
az=0;
el=0;
% i_count = 1;
% flag = ['g'];
% newz = [0,0];
step_az = 15;
step_el = 15;
for i_az = 0:11;
```

```
    for i_el = 0:23;
        az = step_az*i_az;
        el = step_el*i_el;
        disp(['AZ=',num2str(az),', EL=',num2str(el)]);
        view([az,el]);
        disp('Hit any key to continue.');
        pause;
%         if mod(i_count,10) < .5,
%             while mod(i_count,10) < .5,
%                 flag = input('Go or backtrack? (''''g''''
or ''''b''''?)\n');
%                 if flag == ['b'],
%                     newz = input('Give temp
''[AZ,EL]''.\n');
%                     view([az,el]);
%                     flag = ['g'];
%                     i_count = 10;
%                 else,
%                     i_count = 11;
%                 end;
%             end;
%         end;
%         i_count = i_count + 1;
    end;
end;
```

CODE 8

```
%{
Leone, 2006
This code graphs only the random alloy line.
%}

close all; clear all; clc;format short g; delta = .00001;

load('J_vals.data');
f_RA=inline('ao+a1*(2*x-1)+a2*(2*x-1).^2+a3*(2*x-
1).^3+a4*(2*x-1).^4+a5*(2*x-1).^5+a6*(2*x-1).^6',...
    'x','ao','a1','a2','a3','a4','a5','a6');
x=linspace(0,1,100); i_MBIT = 1;
ao = J_vals(i_MBIT,2);
i_MBIT = i_MBIT + 1;
a1 = J_vals(i_MBIT,2); i_MBIT = i_MBIT + 1;
a2 = 0;
while J_vals(i_MBIT,1) < 3,
    a2 = a2 + J_vals(i_MBIT,2); i_MBIT = i_MBIT + 1;
```

```
end;
a3 = 0;
while J_vals(i_MBIT,1) < 4,
    a3 = a3 + J_vals(i_MBIT,2); i_MBIT = i_MBIT + 1;
end;
a4 = 0;
while J_vals(i_MBIT,1) < 5,
    a4 = a4 + J_vals(i_MBIT,2); i_MBIT = i_MBIT + 1;
end
a5 = 0;
while J_vals(i_MBIT,1) < 6,
    a5 = a5 + J_vals(i_MBIT,2); i_MBIT = i_MBIT + 1;
end
a6 = 0;
while J_vals(i_MBIT,1) < 7,
    a6 = a6 + J_vals(i_MBIT,2); i_MBIT = i_MBIT + 1;
end
Eg_RA = f_RA(x,ao,a1,a2,a3,a4,a5,a6);
% A=[ao,a1,a2,a3,a4,a5,a6];


% Defense
set(0,'defaultaxesfontsize',40);
set(0,'defaulttextfontsize',40);
MM = 5;
figure(1);
plot(x,Eg_RA,'b-','linewidth',MM);
title('Predicted Random Alloy Band Gap');
hold on;
xlabel('ZnO Concentration');
ylabel('E_{g} (eV)');
axis([0,1,3,8]);

% Paper
set(0,'defaultaxesfontsize',30);
set(0,'defaulttextfontsize',30);
MM = 3;
figure(2);
plot(x,Eg_RA,'k-','linewidth',MM);
title('Predicted Random Alloy Band Gap');
hold on;
xlabel('ZnO Concentration');
ylabel('E_{g} (eV)');
axis([0,1,3,8]);
```

CODE 9

```
%{
Leone, 2006
This script does a pretty good job of creating
nice band structure plots.  It also determines
indirect band gaps.
%}

textshift = 15; % Places 'Eg = ' text appropriately.

reload = 1; % Use with 1 once, then leave at 0 to adjust
plots.
if reload == 1,
    %%% Modify this first section to change structures.
    close all;clear all;clc;format short g;
    %load a_bands/eig_MgO.data;
    load a_bands/eig_ZnO.data;
    %load a_bands/eig_str1.data;
    %load a_bands/eig_str2.data;
    %load a_bands/eig_str10.data;

    textshift = 15; % Places 'Eg = ' text appropriately.
    titL = ['Non-Stacking ZnO'];%ZnO_{1}-MgO_{3} Primitive
Cell'];
    eig = eig_ZnO;

    %%% End Modifications.

    set(0,'defaultaxesfontsize',30);
    set(0,'defaulttextfontsize',30);
    set(0,'defaultlinelinewidth',1);

    %% Data specific to k-points
    n_kp = 304;
    % 1 L, 76 G, 163 X, 207 W, 304 G ...
    % changing these will force mods below.

    n_bands = eig(1,1);
    Ef = eig(1,2);
else,
    close all;
end;


% Find indirect band gap, report and plot (below).
n_c = [0,100];
n_v = [0,-100];
% cv_swap = [0,0;0,0];
```

```
for i_i = 2:n_kp*n_bands+1,
    if (eig(i_i,2)-Ef < 0 ) && (eig(i_i,2) > n_v(2)),
        n_v(1:2) = [[eig(i_i,1)],[eig(i_i,2)]];
    elseif (eig(i_i,2)-Ef > 0 ) && (eig(i_i,2) < n_c(2)),
        n_c(1:2) = [[eig(i_i,1)],[eig(i_i,2)]];
    end;
end;
% Move to correct gamma.
if abs(n_c(1) - 304) < .5,
    n_c(1) = 76;
end;
if abs(n_v(1) - 304) < .5,
    n_v(1) = 76;
end;
eig_text = ['indirect'];
if abs(n_c(1) - n_v(1)) < .5,
    eig_text = ['direct'];
end;


% Rearrange data into bands.
eig_bands = eig;
i_count = 2;
for i_b = 1:n_bands,
    for i_kp = 1:n_kp,
        eig_bands(i_count,:) = eig(1+n_bands*(i_kp-
1)+i_b,:);
        i_count = i_count + 1;
    end;
end;


% Plot data.
eig_bands(2:end,2) = eig_bands(2:end,2) - Ef;
n_c(2) = n_c(2) - Ef;
n_v(2) = n_v(2) - Ef;
x = linspace(1,n_kp,n_kp);
eig_bands(2:end,2) = eig_bands(2:end,2);
plot(x,eig_bands(2:n_kp+1,2),'k-');
axis([x(1),x(end),min(eig_bands(2:end,2))-1,...
    max(eig_bands(2:end,2))+1]);
set(gca,'xtick',[x(1)-1,x(end)+1])
title(['Band Plot of ',titL]);
ylabel('E_{band} (eV)');
text(x(1)-3,min(eig_bands(2:end,2))-2.5,'L');
text(x(76)-3,min(eig_bands(2:end,2))-2.5,'\Gamma');
text(x(163)-4,min(eig_bands(2:end,2))-2.5,'X');
text(x(207)-6,min(eig_bands(2:end,2))-2.5,'W');
text(x(304)-3,min(eig_bands(2:end,2))-2.5,'\Gamma');
text((n_v(1)+n_c(1))/2 + textshift,(n_v(2)+n_c(2))/2,...
    ['E_{',[eig_text],' gap} = ',num2str(n_c(2)-n_v(2)),'
eV']);
hold on;
for i_pb = 2:n_bands,
    plot(x,eig_bands(2+n_kp*(i_pb-1):1+n_kp*i_pb,2),'k-');
end;
plot([[n_v(1)],[n_c(1)]],[[n_v(2)],[n_c(2)]],...
    'k-','linewidth',4);
plot([76,76],[min(eig_bands(2:end,2))-1,...
    max(eig_bands(2:end,2))+1],'k-');
plot([163,163],[min(eig_bands(2:end,2))-1,...
    max(eig_bands(2:end,2))+1],'k-');
plot([207,207],[min(eig_bands(2:end,2))-1,...
    max(eig_bands(2:end,2))+1],'k-');
% EF = Ef*ones(1,n_kp); % I have now shifted Ef to zero
% text(x(5),EF(1)+(max(eig(2:end,2))-
min(eig(2:end,2)))/25,...
%     'E_{f}'); % I have now shifted Ef to zero
% plot(x,EF,'k--','linewidth',1); % I have now shifted Ef
to zero
hold off;
```