Ab-initio investigation of the elastic behavior of the ternary compounds $C_xSi_yGe_{1-x-y}$ in the cubic phase

MASTERARBEIT

zur Erlangung des akademischen Grades
Master of Science
(M. Sc.)
im Fach Physik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät I
Institut für Physik
Humboldt-Universität zu Berlin

von
Herrn Tim Klose
geboren am 23.2.1990 in Berlin

Betreuung:

1. *Prof. Dr. Dr. h. c. Claudia Draxl*
2. *PD Dr. Pasquale Pavone*

eingereicht am: *22. August 2018*

Dedication

“Für nichts und wieder nichts.”

Contents

1 Introduction **1**

2 Theoretical background **3**
2.1 Density functional theory 3
2.1.1 How density functional theory works 3
2.1.2 The exchange-correlation energy functional 4
2.1.3 (L)APW+lo 4
2.2 Calculating elastic constants 6
2.2.1 The theory of elastic constants 6
2.3 `ElaStic` and `exciting` 7

3 Investigated configurations **9**
3.1 Making ab-initio choices for our structures 9
3.2 Nomenclature 9
3.3 Finding geometrically inequivalent configurations 10
3.4 Choosing which structures to investigate 11
3.5 Creating a starting point for numerical calculations 12

4 Results: Lattice constants **13**
4.1 Summary of our results 13
4.2 Analysis of the lattice constants 13
4.2.1 The "scalar product" 13
4.2.2 The prevalence of bond breaking 15
4.3 Predicting lattice constants 16

5 Results: Elastic constants **18**
5.1 Summary of our results 18
5.2 Analysis of the elastic constants 19
5.3 Predicting elastic constants 20

6 Conclusions and outlook **22**

7 Appendix **25**
7.1 Used species files 25
7.2 Matlab code to find configurations 26
7.3 List of configurations, structural properties, lattice constants and bulk and shear moduli 31
7.4 List of cumulated elastic constants 33
7.5 Convergence tests 34
7.6 Glossary/Abbreviations 37

List of Figures

2.1 The two main types of strain. 6

3.1 Nomenclature example for the system 1112.1233. Green atoms are C, red is Si and blue is Ge. 10
3.2 A schematic view of our investigated chemical compositions. 11

4.1 The different structures of the chemical composition Ge_4C_4. 14
4.2 Side-by-side comparison of systems with and without broken bonds. . 15
4.3 Carbon-carbon distances of nearest neighbors for different structures. 15

5.1 Bulk and shear moduli before correcting for a^{-4}. 19
5.2 Bulk and shear moduli after correcting for a^{-4}. 20

Chapter 1

Introduction

The motivation behind this thesis was to comprehensively map the elastic constants (and some other properties that are accrued along with them) of materials containing carbon (C), silicon (Si) and germanium (Ge) in a bulk environment.

One perspective use of this data is to make predictions about larger and more complex systems formed with these elements. One particularly interesting topic will be to see how the individual atoms can form microscopic substructures that affect the macroscopic physical properties. It will also be interesting to see how mixing differently sized atoms in one crystal will affect their bonding behavior, again with major implications for the elastic constants.

All of the structures that we are going to investigate are derived from the diamond structure. This is the hardest configuration for all three elements; it is also the most thermodynamically stable for pure silicon and germanium as well as silicon carbide and silicon-germanium at ambient conditions; the diamond structure of carbon is metastable [8]. Germanium carbide is always a metastable compound (its lowest formation energy in our testing was +71 kJ/mol), but the diamond structure is again the most stable.

These types of compounds feature in such a huge array of technical appliances, so we are only going to mention the ones that are most relevant for our work.

Their most common physical feature is hardness, which ranges from pure germanium - the softest compound in our calculations, but still about as hard as steel - to diamond and silicon carbide (SiC), which are among the hardest known compounds [8]. Both are used as an abrasive and to manufacture tools.

The compounds do differ greatly in their electrical behavior. Diamond is an insulator, while pure silicon and germanium are semiconductors, and have long been used as such in transistors and other electrical components. While we did not investigate conductivity explicitly, we do have data for the density of states at the Fermi energy as calculated by **`exciting`**, which should allow to us to make some general remarks and predictions about the electrical behavior of the various configurations.

Another aspect of this work is big data analysis, both in the context of finding noteworthy patterns in the data we already generated, as well as providing our data to others for big data analysis with an even larger sample size.

In this work, we start by setting up an efficient nomenclature for carbon, silicon and germanium in a diamond structure in an 8-atom cubic unit cell. Next, we find out which configurations are geometrically inequivalent, determine which structures to investigate and find sensible parameters for our computation. With all of the lattice and elastic constants calculated, we analyze the results and try to find patterns. Since one of the goals of our work is the ability to predict lattice and elastic constants in larger systems, we especially focus on their correlation to structural properties that we can (in hindsight) determine ab-initio.

Chapter 2

Theoretical background

In this work, we are going to employ the **`exciting`** [4] package to calculate the energy of a crystal for a given nuclear configuration. Using these energies, we will employ the **`ElaStic`** package to calculate the lattice constants and the second-order elastic constants.
In this chapter, we will first outline the workings of density functional theory, followed by the specific implementation in **`exciting`** with the LAPW+lo basis set.

We will continue with a short introduction to second-order elastic constants, and then showcase how the **`ElaStic`** package [7] does calculate them from the energies provided by an **`exciting`** calculation.

2.1 Density functional theory

Density functional theory (DFT) was developed as an alternative approach to solve the Schrödinger equation. It is based on the Hohenberg-Kohn theorem, which established that any given ground-state (GS) electron density distribution has only one corresponding (external) potential. Thus the knowledge of the GS electron density theoretically allows the evaluation of all the properties of a system [6].

Calculating the electron density instead of the wave functions is hugely advantageous for computing, since the computation time scales only with the third power of the particle number, rather than having an exponential relation [4].

In our calculations, the external potential will be solely generated by the atomic cores, but a further manipulation of the potential would also be possible to simulate all kinds of external physical interactions.

2.1.1 How density functional theory works

As a consequence of the Hohenberg-Kohn (HK) theorem, expectation values for an arbitrary operator $\hat{O}$ can be expressed as a functional of the N-particle (electron) GS density (first theorem):

$$\langle\Psi|\hat{O}|\Psi\rangle = O[\rho_{GS}] \tag{2.1}$$

A corollary of the HK theorem states that if the functional $E_V[\rho] = H[\rho]$ reaches its minimal value, ρ is the correct GS density corresponding to the potential V

(variational principle).

With this theorem, the physical foundation of DFT is set. Now we need to find an appropriate GS energy functional (including an exchange-correlation functional) and a solving algorithm. The latter was found by Kohn and Sham (KS) with the introduction of a fictitious system of non-interacting electrons, which are described by a single-particle Schrödinger-type KS-equation:

$$\hat{h}_{KS}\psi_i(\mathbf{r}) = \epsilon_i\psi_i(\mathbf{r}) \tag{2.2}$$

where the KS Hamiltonian is (in atomic units)

$$\hat{H}_{KS} = -\vec{\nabla}^2 + \int \frac{\rho(\mathbf{r'})}{|\mathbf{r} - \mathbf{r'}|}d\mathbf{r'} + \hat{\mathrm{v}}_{\mathrm{xc}}(\mathbf{r}) + \mathbf{v_{ext}}(\mathbf{r}). \tag{2.3}$$

In Eq.˜(2.3), the first two terms are the kinetic energy and the Hartree potential, followed by the exchange-correlation potential, now given by

$$\hat{v}_{xc}(\mathbf{r}) = \frac{\delta E_{xc}[\rho]}{\delta\rho(\mathbf{r})} \tag{2.4}$$

The KS theorem states that the N lowest-energy solutions of the KS equation are wavefunctions for virtual particles, which then add up to the correct physical particle density in the usual way:

$$\rho(\mathbf{r}) = \sum_{i=1}^{N} \psi_i^*(\mathbf{r})\psi_i(\mathbf{r}) \tag{2.5}$$

With this ρ-dependent KS Hamiltonian, the KS equations can now be solved in a self-consistent calculation. In doing so, we have obtained a tool for the computational implementation of iterative DFT ground-state calculations.

2.1.2 The exchange-correlation energy functional

The only thing left before the actual calculation is to specify a suitable exchange correlation functional. The two most important groups of functionals are named *Local-Density Approximations* (LDA) and *Generalized-Gradient Approximations* (GGA). The LDA functional only includes terms that depend on $\rho(\mathbf{r})$ itself. These are derived from the solution of the homogenous electron gas (HEG). In this case, the exchange correlation functional assumes the following form in the LDA:

$$E_x^{LDA}[\rho] = -\frac{3}{4}\left(\frac{3}{\pi}\right)^{1/3} \int \rho(\mathbf{r})^{4/3} d\mathbf{r} \tag{2.6}$$

GGA functionals also include terms using spatial derivatives of $\rho(\mathbf{r})$. This does make GGA-functionals more flexible. There are also functionals that go beyond the GGA, like the exact exchange functional. However, in this work we are exclusively going to use the LDA functional.

2.1.3 (L)APW+lo

Now, if we want to find the (so far) completely unknown KS-wavefunctions, we have to find a basis set to express them. Since **`exciting`** is dealing with periodic unit cells, the obvious first choice are Bloch-periodic plane waves:

$$\phi_i(\mathbf{r}) = \sum_{\mathbf{G}} \frac{1}{\sqrt{\mathrm{V}}} \mathrm{e}^{\mathrm{i}(\mathbf{k}+\mathbf{G})\cdot\mathbf{r}} \tag{2.7}$$

Plane waves are a good choice to describe particles with a large spatial volume such as valence and conduction band electrons. However, core electrons are much smaller, and they would require a number of plane waves that are inconvenient for computation purposes to achieve an accurate description. The simplest way to get around this is to treat the core electrons independently with numerical solutions of the spherical Schrödinger equation (since the inner wavefunctions are almost orthogonal), and treat the outer electrons with plane waves. This is known as the *pseudopotential* method.

More accurate codes like **`exciting`** take the next step up with the introduction of *Augmented Plane Waves* (APW). Methods with this name all divide the unit cell into spherical “muffin-tins” (MT) around the atomic cores and the interstitial space (I) in between. For I, the APW remain the same, but for MT, they are now given as follows:

$$\phi^{\mathbf{k}}_{\mathbf{K}}(\mathbf{r},\mathrm{E}) = \sum_{\mathrm{l,m}} \mathrm{A}^{\alpha,\mathbf{k+K}}_{\mathrm{lm}} \mathrm{u}^{\alpha}_{\mathrm{l}}(\mathrm{r}',\mathrm{E})\mathrm{Y}^{\mathrm{l}}_{\mathrm{m}}(\hat{\mathrm{r}'}) \quad \text{for} \quad \mathbf{r}\in \mathrm{MT}, \tag{2.8}$$

where $\mathbf{r}'$ is the spatial coordinate relative to the corresponding atomic core / the center of the muffin-tin. The Y^l_m are spherical harmonics, the u^α_l are solutions to the radial part of the spherical Schrödinger equation for the free atom. All the terms are **k**-dependent, while the A-parameter has to be chosen in such way that the APW is continuous at the border of MT and I. The parameter that remains now is E. Since it is both an input parameter and also the output energy, once the KS-wave function is constructed from all APWs, an initial guess has to be made for E, which is then converged to the right value by comparing input and output energy. APWs are better matched to the real structure of the KS wavefunctions, and they require less basis vectors (APWs) than pseudopotential plane waves. However, they do require one diagonalization for every KS-eigenvalue due to the energy search and are therefore slower than pseudopotential methods.

The pure APW method is now improved further in two different ways.
The first is called *Linearized Augmented Plane Waves* (LAPW). To circumvent the trial-and-error method of searching the energy eigenvalue in APW, it starts with an educated guess for the energy eigenvalue of the particle ϵ_n (called E_0) and Taylor-expands u^α_l around it:

$$u^\alpha_l(\epsilon_n) = u^\alpha_l(E_0) + (\epsilon_n - E_0)\frac{\partial u^\alpha_l(E)}{\partial E}|_{E=E_0} \tag{2.9}$$

Terms of higher order are neglected. This expression for u^α_l is entered into the muffin-tin part of the original APW (where we need a new parameter B for the yet unknown $\epsilon_n - E_0$). It is now possible to set up a single self-consistent equation and solve directly for ϵ_n. This is why LAPW is faster than APW, even though it needs slightly more plane waves to achieve the same accuracy.

The second method for improving APW is called $APW + lo$ (“lo” stands for local orbitals). The idea behind local orbitals is to add even more terms to the muffin-tin wave function. To keep the rest of the APW the same, they are required to reach a value of zero at the edge of the muffin-tin sphere to retain the APW’s continuity. Like the LAPW method, this method now only requires a single diagonalization, while retaining the same accuracy as pure APW.

`exciting` uses a combination of both LAPW and APW+lo to employ a mixed LAPW/APW+lo basis set (dubbed "(L)APW+lo"). Therein, APW+lo is used for valence d- and f-states as well as states in atoms with small muffin-tin-radii, while LAPW is used for the remainder of the states. This combination, according to current knowledge, is the most efficient way to calculate the groundstate density in a DFT calculation with a proper treatment of the interaction between all electrons[4]. In **`exciting`** ground-state calculations, the maximum wave vector is set by the parameter $R_{MT} \cdot G_{max}$, where R_{MT} is the smallest MT-radius defined in the system. If $R_{MT} \cdot G_{max}$ is chosen too high, the wave functions will be overdetermined by the basis set, which results in a diverging calculation.

2.2 Calculating elastic constants

2.2.1 The theory of elastic constants

The simplest example of an elastic constant is featured in Hooke's law:

$$\sigma = C \cdot \epsilon \tag{2.10}$$

Hooke's law describes an ideal spring, where the stress σ and the strain ϵ are proportional, in the one-dimensional case; all the values involved are thus scalars.
The proportionality of stress and strain is obviously far from ubiquitous in the real world, but Hooke's law will always be applicable in a small region around the equilibrium known as the *elastic linear regime*; special cases like phase transitions near the equilibrium are excluded from this rule.

To generalize Hooke's law from one to three dimensions, we have to replace the scalars with tensors. First, we are going to look at the strain to find the different types of possible distortions. We are going to use cubes as our initial shape, since it facilitates the explanation (and does represent our unit cells), but the principles apply to other shapes as well. If we align the edges of the cube along the axes in Euklidian space, two opposite sides of the cube are perpendicular to the x-axis. Now, there are two fundamental ways in which we can introduce strain into our cube:

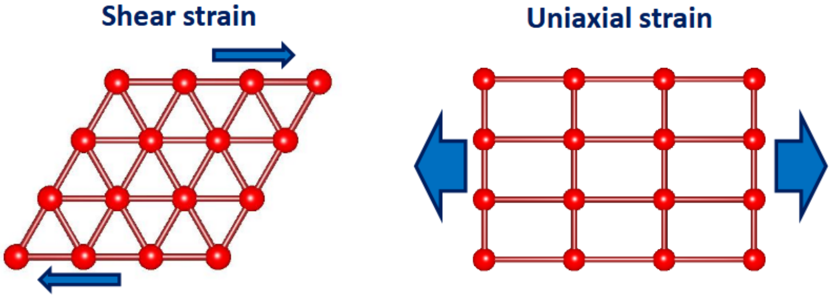

Figure 2.1: The two main types of strain.

The distortion on the left is called "shear strain", while the distortion on the right is called "uniaxial" or "normal strain". In three dimensional space, there is just one type of normal strain for each direction (ϵ_{ii}), while shear strain has two possibilities (ϵ_{ij} and ϵ_{ik}). However, we can see that for small distortions, there is no distinction between ϵ_{ij} and ϵ_{ji}, which reduces the total number of linearly independent strains in three dimensional space to 6, these are (in standard notation and order) ϵ_{xx}, ϵ_{yy}, ϵ_{zz}, ϵ_{yz}, ϵ_{xz} and ϵ_{xy}. In the *Voigt notation* these strains are renamed ϵ_1, ϵ_2, ϵ_3, ϵ_4, ϵ_5 and ϵ_6, which reduces their total dimension from 2 to 1. The shear strains are multiplied by two to keep energy calculations consistent, since they are technically two strains morphed into one:

$$\epsilon_4 = \epsilon_{yz} + \epsilon_{zy} = 2 \cdot \epsilon_{yz} \tag{2.11}$$

Next, we move on to the stress tensor σ. With the same reasoning as for the strain tensor, we again find that it has 6 independent components. They are renamed the same way in the Voigt notation (e.g. $\sigma_{xx} == \sigma_1$), even though there is no pre-factor this time for the transversal components, since they were already added to the strain ($\sigma_4 = \sigma_{yz}$, etc.).

With this new notation for stress and strain established, we can now write a three dimensional version of Hooke's law as follows:

$$\sigma_\alpha = C_{\alpha\beta} \cdot \epsilon_\beta \tag{2.12}$$

This new tensor **C** does have 6 x 6 = 36 components. However, there are a number of ways to show that **C** has to be symmetrical as well ($C_{ij} = C_{ji}$), which leaves a maximum of 21 independent components, meaning that any system will have a maximum of 21 second order elastic constants [7].

2.3 `ElaStic` and `exciting`

The algorithm we use to evaluate the elastic constants for our system is **`ElaStic`**, which is by default included (and fully implemented) in the **`exciting`** package.

`ElaStic` determines the internal geometry using the function sgroup, which assigns each structure with their respective geometry group. It does allow for small deviations from the theoretical (perfect) atomic positions to find the highest possible symmetry with the lowest number of independent elastic constants. Depending on the geometry, **`ElaStic`** then sets up the same number of (linear) distortions. **`ElaStic`** can not evaluate stress as such, since **`exciting`** does not provide external forces as output. It rather applies a certain kind of strain to the unit cell, and then another linearly independent strain to evaluate the stress along this direction using the resulting total energies, since energy is the integral of stress over strain. This is possible since the tensor **C**, which relates stress and strain, is symmetrical.

`ElaStic` distributes the distortion data points with different amounts of strain symmetrically around the undistorted structure to allow for a better interpolation of asymmetric energy curves, and to potentially compensate for initial structures that are not fully relaxed. **`ElaStic`** automatizes this step by creating and prearranging these calculations. Each calculation is a basic **`exciting`** groundstate calculation, yielding the respective total energy as output.

It is up to the user to pick the set of data points along with the degree of polynomial to compute the elastic value for each distortion, which can affect the result somewhat for the imperfect energy-vs-strain relations encountered in realistic scenarios. These do often, but not always, result from some kind of discernible phase transition. We used 9 data points with a maximum strain of 0.04, and we used a polynomial of grade 4 to approximate the stress-vs-strain data table produced by every single distortion.

With this input given, **`ElaStic`** proceeds to calculate the elasticity tensor, along with the derived values for different types of bulk moduli, shear moduli and poisson ratios.

Chapter 3

Investigated configurations

In this chapter we are going to talk about finding the configurations that we are going to investigate, and the reasoning and the methods applied to find the starting point for the calculation of lattice and elastic constants.

3.1 Making ab-initio choices for our structures

The most major decision we made before starting calculations was to use cubic unit cells with 8 atoms. To start, including 8 instead of the basic 2 atoms of the diamond structure allows us to analyze structures with more than 2 atomic species in it. Including more than 2 atoms in a unit cell for diamond-type structures also allows internally shifted structures, where the bond lengths between two atoms deviate from the distance these atoms would have in an undisturbed environment, which will produce a number of noteworthy effects.

We also chose to keep a perfect cubical shape for all structures and chemical compositions, even if it does not represent the energetic equilibrium for that structure. By doing so, we implicitly assume that these unit cells analyzed by us will serve as part of a bigger crystal that contains a number of geometrically different unit cells, where the forces exerted by the single unit cells will combine to force each unit cell (roughly) into a cubic shape, as well as forming a crystal of cubic geometry. Other choices such as a trigonal or triclinic unit cell would have been possible (and under different circumstances sensible) as well, but a cubic unit cell is arguably the least arbitrary unitary choice, and it maximizes comparability in between configurations.

3.2 Nomenclature

Since we are going to be dealing with a lot of structures, we need an effective nomenclature to identify individual ones. Our first step was to swap the names carbon, silicon and germanium for the numbers 1,2 and 3 (in that order) to shorten the resulting names. Next, we take geometry into account. The primitive unit cell for diamond cubic structures contains two inequivalent atomic sites, which may or may not be occupied by different atoms. If they are, each atom has 4 nearest neighbors of the opposite species in the crystal.

If we build a cubic unit cell with 8 atoms, we hence get 2 groups of 4 geometrically equivalent atoms each (in undistorted structures). Again, each atom will be neighbored by 4 geometrically equivalent atoms, which are all occupying different positions in the cubic unit cell.

Due to this geometry, we choose to list one group of geometrically equivalent atoms in the first 4 digits and the other group in the last 4, and separate them by a point for better legibility. Inside the cubic unit cell, both groups form a tetraeder with the same orientation and edge length, with a shift of (0.25 0.25 0.25)a in between them. To account for this circumstance, the digits are arranged in such a way that the first atom's position shifted by this vector equals the fifth atom's position, the second goes to the sixth and so on. One practical example for this nomenclature:

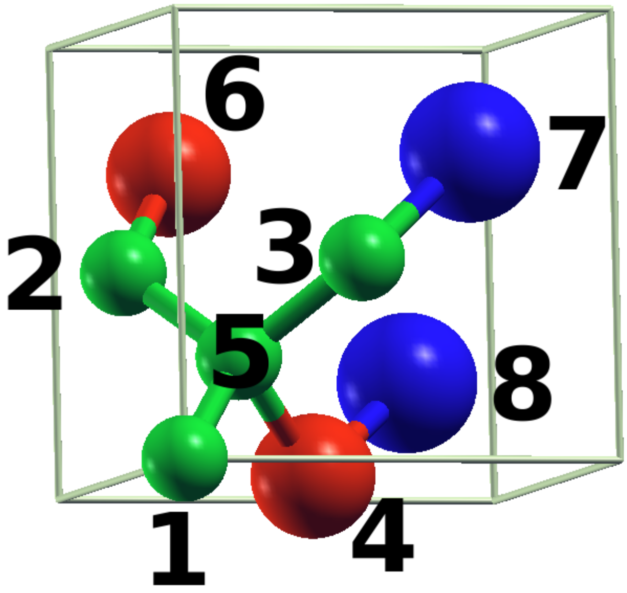

Figure 3.1: Nomenclature example for the system 1112.1233. Green atoms are C, red is Si and blue is Ge.

At last, the so-named configurations are put into 'alphabetical' order, starting with 1111.1111, then 1111.1112 and finishing with 3333.3333. If two configurations are geometrically equivalent, the name that comes first in this order is chosen to describe the structure.

3.3 Finding geometrically inequivalent configurations

With our ab-initio choices made, we now have to proceed with finding the geometrically inequivalent configurations structures that can be formed with 8 atoms of the species C, Si, and Ge in a diamond structure. We start with the $3^8 = 6561$ possibilities of filling 8 atomic sites with 3 types of atoms and subsequently reduce for translational, rotational and mirror symmetry by using a Matlab script.

The basic idea behind the script is to set up a configuration catalog ranging from 0 to 6560. Starting with 0, we convert each entry into a 3D-matrix - resembling the actual geometry of a cubic diamond unit cell - using a set algorithm, and then search the 3D-matrix from all (rotational) directions and positions to find appropriate structures, for which we assign a number by reversing the structure-building algorithm. All the entries corresponding to the so-calculated numbers (except for the original one) are then eliminated from the catalog, and the number of the equivalent configurations is recorded.

After all this is said and done, the original 6561 configurations have been reduced to 141 geometrically inequivalent configurations, each of them representing anywhere from 1 to 192 of the original configurations.

3.4 Choosing which structures to investigate

Since we have 3 types of atoms to choose from, we can either visualize all possible chemical compositions as a plane (since the total number of atoms is constant) in a three-dimensional grid or (more conveniently) project this plane into a two-dimensional triangle:

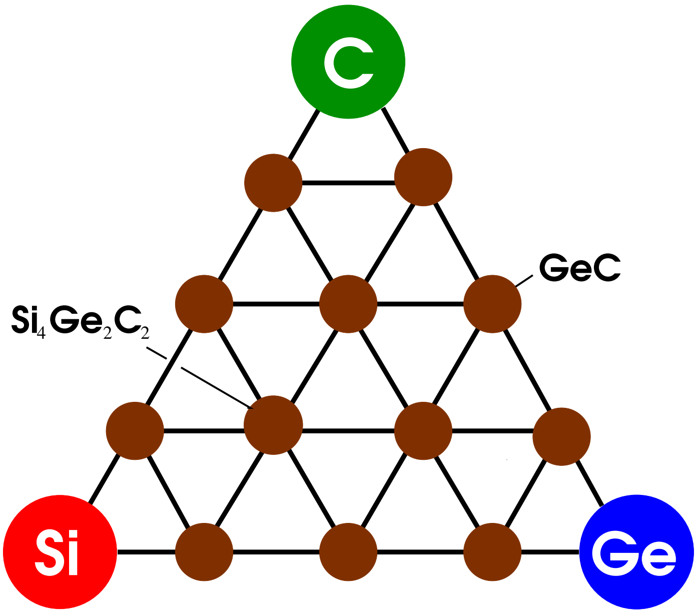

Figure 3.2: A schematic view of our investigated chemical compositions.

The corners of the triangle represent the pure compounds carbon, silicon and germanium. The edges of the triangle thus represent binary compounds that contain the two elements of the corners they connect, with a varying proportion of the two species depending on the position. The inside of the triangle contains all the configurations which contain all three atomic species. Out of the 141 configurations, we have the three pure elements, 45 binary configurations and 93 ternary configurations. There are only 45 possible chemical compositions for a 3-species 8-atom unit cell (3 pure, 21 binary, 21 ternary), thus every chemical composition can divide into 1 to 7 geometrically inequivalent configurations. The number is lowest towards the corners of the triangle, while the ternary chemical compositions in the middle - which have at least 2 atoms of every species - always feature 7 inequivalent configurations.

Since many of the 141 configurations are quite similar in character, we decided that it is reasonable to make the grid a bit coarser and exclude configurations with odd numbers of a certain species. In the triangle, all the chemical compositions that are included are marked with colored points, while the excluded compositions lie in the middle of the lines connecting these points. This retains all the extremes of possible configurations - we still include pure C, Si, Ge, SiC, GeC and SiGe, as well as ternary compounds with 7 different configurations for a given chemical composition.

This results in a reduction of investigated configurations from 141 to 48, but since most of the eliminated configurations are ternary, total computation time is reduced by more than 75%. Our dataset was still more than large enough to find all kinds of meaningful conclusions, which makes it unlikely that the exclusion of these configurations had any detrimental effect on the results.

3.5 Creating a starting point for numerical calculations

Once we completed the selection of the configurations we would investigate, we had to come up with a method of turning these configurations into geometrical data that we can enter into a numerical computation (= creating the structure).

After some consideration, we eventually decided to use the basic undistorted diamond structure as a starting point. We also had to choose a starting lattice constant which was close to the equilibrium. We started bona fide with the value provided by Vegard's law (dubbed ¨Vegard lattice constant¨, VLC), which is proportional to the content of a certain kind of atom as well as their respective (pure) lattice constants, and generally a well-established (and logical) method for these kind of purposes:

$$VLC = x_C \cdot a_C + x_{Si} \cdot a_{Si} + x_{Ge} \cdot a_{Ge} \tag{3.1}$$

x_i is the relative content of each species. This provided us with a useful starting point for our calculations. We fully relaxed all of the configurations internally before proceeding to calculate lattice or elastic constants to minimize the required computation time.

Chapter 4

Results: Lattice constants

In this chapter, we are going to present our results for the lattice constants and do some analysis of their behavior. We will then use that to craft a formula to predict them. With all of the configurations set to their respective VLCs and with their internal coordinates relaxed, we used the respective `ElaStic` tool to set up their volume optimizations and used the routine pbirch (employing the Birch-Murnaghan equation of state [2]) to find their equilibrium lattice constant a.

4.1 Summary of our results

The full list of *calculated lattice constants* (a) is to be found in the appendix.

As was to be expected, pure germanium had the biggest lattice constant of all, while pure carbon had the lowest. In general, the behavior of the lattice constants was rather closely matched to their VLCs; the biggest outlier was 1133.1133. where a exceeded VLC by 0.295 Bohr or about 3.5%. Pure SiC (1111.2222) had the lowest lattice constant compared to its VLC, the VLC being 0.259 Bohr or about 3% higher. In general, configurations with a lot of silicon tended to have slightly lower a compared to the VLC, while configurations with lots of germanium tended to have higher ones.

4.2 Analysis of the lattice constants

With the finished results for the difference of the lattice constants in comparison to the VLC, we systematically searched for meaningful correlations between this difference and the corresponding geometric configuration. This includes the number of the different types of bonds (e.g. the comparably weak Ge-C bond) and other quantifiable geometric features. Besides the Ge-C bonds, we actually reduced ourselves to one geometric feature which we denominated the 'scalar product' (s), since no other discernible feature showed a promising correlation to the physical properties.

4.2.1 The "scalar product"

In geometric terms, s represents the number of geometrically inequivalent points (out of a maximum of 4) in the 111-plane that are exclusively occupied by the larger atoms Si and Ge. In our nomenclature of the configurations, we compute s by assigning the small atoms (C) with a '0' and the larger atoms (Si, Ge) with a '1', e.g.

turning the configuration '1123.1213' into 0011.0101. We now multiply the atoms from the first group of geometry-equivalent atoms with the related atoms of the second group, and the sum of those is the scalar product (in this case it would be $s = (0011) \cdot (0101) = 0 \cdot 0 + 0 \cdot 1 + 1 \cdot 0 + 1 \cdot 1 = 1$).

To illustrate the effect of s visually, we chose the 4 possibles structures with the chemical composition Ge_4C_4, which assume s values ranging from 0 to 2. All structures are shown in a 2x2x2 unit cell arrangement from the same angle:

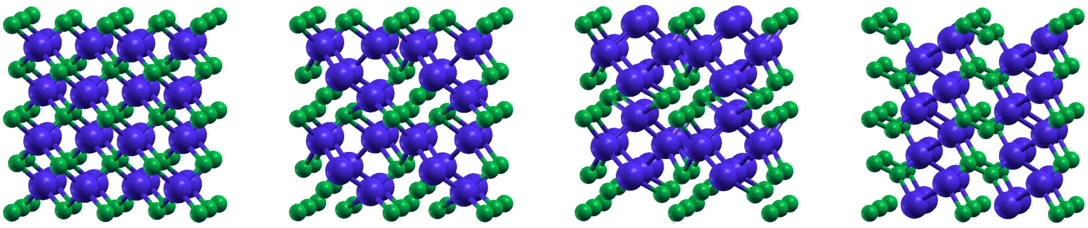

Figure 4.1: The different structures of the chemical composition $\mathrm{Ge}_4\mathrm{C}_4$.

The structure 1111.3333 exhibits no shifts whatsoever compared to a perfect diamond structure, each atom is bordered by 4 atoms of the other species and there are no discernible substructures. This is (or be it with slight distortions) the case for any $s = 0$ structure.

The structures 1113.1333 and 1133.1313 have $s = 1$. These are on paper quite different, since 1113.1333 has atoms bound to 1 OR 3 atoms of the other species, while in the structure 1133.1313 they are always bound to 2 atoms of the other species. However, they are visually quite similar in that they form localized substructures.

The structure 1133.1133 has $s = 2$. This structure forms extended alternating planes of germanium and carbon.

Geometrically speaking, a higher s does indicate a more anisotropic crystal, since the there is a higher mismatch amongst the equivalent atoms in our unit cell; e.g. in 1111.3333 (s=0) all of the geometrically equivalent atoms are of the same species, while in 1133.1133 both groups are a mix of two C and two Ge atoms. One caveat is that for perfectly isotropic materials like pure silicon (2222.2222), s is still 4. This is highly impractical for any prediction formula, since for pure silicon (and germanium), a is exactly the same as the VLC.

Since the VLC is supposed to describe an "average configuration", we now modify our parameter s by substracting an "average s" s_{av}: For structures with zero Si and Ge atoms, s is always 0, so s_{av} is 0, for structures with 2 Si or Ge, s can be 0 or 1, so s_{av} is 0.5, for structures with 4 Si or Ge, s can be 0, 1 or 2, so s_{av} is 1, for 6 Si or Ge, s is 2 or 3, so s_{av} is 2.5 and for configurations without carbon $s = s_{av}$ is always 4. Our modified s is now

$$s_{mod} = s - s_{av} \tag{4.1}$$

and it is listed in the appendix. It can be -1, -0.5, 0, 0.5 and 1 in our configurations. This may seem like a rather crude and somewhat elaborate approach, but it works very well empirically - as we will see.

4.2.2 The prevalence of bond breaking

Broken bonds are going to become very important for the analysis of the elastic constants, but we will already discuss them here since they are closely related to the parameter s_{mod}.

Before we start with any analysis of the concept of broken bonds, we are going to compare pictures of 1122.1122 and 1133.1133 - the s=2 form of SiC and GeC, respectively - taken from the same angle:

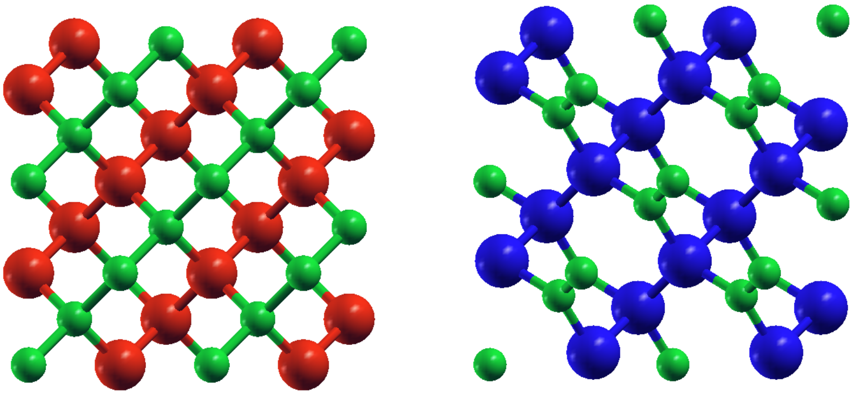

Figure 4.2: Side-by-side comparison of systems with and without broken bonds.

As we can see, half of all C-C bonds that exist in the left structure (1122.1122) are broken in the right one (1133.1133). This behavior can be explained by the fact that the lattice constant for pure C is much smaller than for Si or Ge - 6.675 Bohr compared to 10.214 and 10.644 Bohr, respectively. Since the structure is otherwise the same, unstrained Si-Si and Ge-Ge bonds are more than 50% longer than a C-C bond. Due to this, in a mixed structure, the C-C bond can be put under heavy strain and break. To illustrate this, we counted and ordered all C-C bonds found in our structures by length:

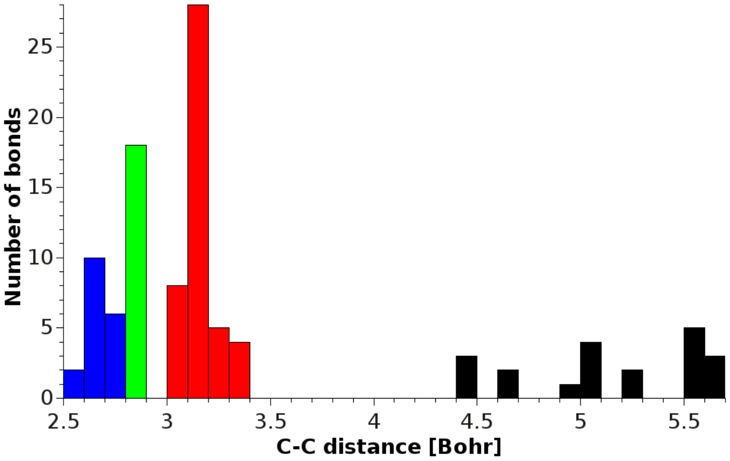

Figure 4.3: Carbon-carbon distances of nearest neighbors for different structures.

As we can see, the bonds can be ordered into 4 categories by their length: Green are the unstrained C-C bonds in pure diamond, with a length of 2.89 Bohr. Red are C-C bonds in mixed structures, which are strained, but not broken, and have lengths ranging from 3.09 Bohr (1111.1122) to 3.37 Bohr (1122.1122). After that, there is a huge gap. The broken bonds have C-C distances ranging from 4.42 Bohr (1112.1112) to 5.64 Bohr (1223.1232). As one would expect for these detached bonds, the C-C distance is roughly proportional to their lattice constants. We can further verify that the bonds are indeed broken by looking at the remaining three bonds for the involved carbon atoms; the angle in between them was always between 119.8 and 120 degrees. An average angular sum of around 359.7 degrees shows that these bonds are in fact located in a plane and that the carbon atoms have switched from an sp^3- to an sp^2-hybridized state [1].

The density of states at the Fermi energy calculated by **`exciting`** is also noticeably higher for these structures, which could be an indicator that these sp^2-hybridized carbon atoms do indeed generate free electrons similarly to graphite [8].

If C-C bonds break in a structure, it does always affect one bond per carbon atom with a C-C bond. If a structure (e.g. 1333.1333) only has one C-C bond, this one bond will break. In a structure like 1133.1133 (see above), every carbon atom has two other carbon atoms as its nearest neighbors, and only one of the bonds will break, while the others remain intact.

Since the remaining bond is now between two carbon atoms in an sp^2-hybridized state, the unstrained bond becomes shorter (about 2.66 instead of 2.89 Bohr)[1]. Accordingly, the bonds in blue on the left are all of this type - since they are part of a structure that also has broken bonds - and their length varies from 2.58 to 2.81 Bohr.

As it turns out, whether bonds will break in a structure or not can be predicted directly from their calculated s_{mod} parameter. If s_{mod} is greater than zero, the bonds will always break (in the pattern we discussed), while they stay intact otherwise.

This is universally true (including strained structures and such) for 46 out of our 48 configurations. In layered SiC (1122.1122, s_{mod}=1), the bonds do not break in the unstrained crystal (probably due to the high symmetry, which distributes the stress evenly among the C-C bonds, preventing them from breaking), but they do break as soon as any strain is applied. This gives this structure an unusually low lattice constant below VLC and one of the lowest shear moduli of all structures. Due to this, we will universally consider this structure as an outlier from now on. The same is true for the structure 1113.1223, where one of the C-C bonds is broken, even though s_{mod} is just 0. In contrast to the previous example, this is probably due to the very low symmetry of this structure, where the two Ge atoms are aligned and the strongest possible atomic mismatch of C and Ge atoms exists in both groups of geometrically equivalent atoms.

4.3 Predicting lattice constants

With all the parameters discussed in the previous section, we are now going to create a formula for the lattice constants. The method we will be using is very straightforward, simply taking the parameters VLC, s_{mod} and n (the number of Ge-C bonds) and varying the coefficients for s_{mod} and n until the difference between formula and a assumes a minimum (if we sum up the absolute values for all configurations). With this method applied we get the following *Formula-generated lattice constant* (FLC) in Bohr:

$$FLC = VLC + 0.245 * s_{mod} + 0.009 * n \tag{4.2}$$

While the average difference between a and VLC was about 0.10 Bohr or 1.1%, the average difference between a and FLC is just 0.04 Bohr or about 0.4 %.

As discussed above, 1122.1122 is the one big outlier with an FLC of 8.69 Bohr and a lattice constant of just 8.39 Bohr, a difference of 0.3 Bohr. No other structure has more than 0.11 Bohr in this measure.

The relatively high prefactor of s_{mod} does justify its introduction, as the term with s_{mod} alone reduces the total error of the formula by more than half.

Chapter 5

Results: Elastic constants

In this chapter, we are going to talk about our results for the *elastic constants* (EC). We are going to show their level of correlation to the lattice constants we discussed in the previous chapter. We are also going to provide an approximate formula for the bulk and shear moduli (B_0 and G) of a given system, and explain why this is so difficult to do.

5.1 Summary of our results

Since our unit cells are always cubic, and since there is no preferred direction of the internal structure, we will always cumulate elastic constants of the same type (=treating the x, y and z-direction as logically equivalent).

We will use their arithmetic mean (or the arithmetic mean of their absolute value for the ones that can be negative), and name them with a capital C and an index denoting their type: "nn" for normal stress and strain along the same direction, "nnp" for normal stress and strain in perpendicular directions, "ss" for shear-shear constants in the same direction, "ssp" for perpendicular ones, and "ns" for elastic constants connecting normal and shear stress and strain.

The full list of cumulated elastic constants is to be found in the appendix.

The elastic constants generally exhibit an inverse behavior to the lattice constants; structures rich in carbon generally have the highest ECs, while structures rich in germanium have the lowest.

When talking about ECs, we have to consider that the different types of elastic constants are not created equal. C_{nn} and C_{ss} describe stresses and strains that are parallel to each other, so they cannot become zero in a solid. C_{sst} and C_{ns} on the other hand describe stresses and strains along different directions, so these two can become zero in a solid and do so for about half of our configurations.

C_{nnt} is a bit of an outlier in this respect, since stress and strain do have perpendicular directions. But since the absence of a transversal response to a longitudinal strain would indicate a poisson ratio of zero, this value will also not be zero for any of our configurations.

In our investigations, the configurations where C_{sst} and C_{ns} are not zero are generally the ones with a lesser symmetry. However, their values are comparatively small (the highest value is just 30.5 GPa), and a value that is zero for half of our configurations is not really useful for analytical purposes and comparisons, so they will be ignored going forward. What we are going to use instead are the bulk and shear modulus B_0 and G, which are combinations of C_{nn}, C_{ss} and C_{nnt} and arguably the most important elastic parameters for real world applications. While there are different ways to calculate the bulk and shear modulus - e.g. the Reuss moduli average the inverse of the ECs - we are going to use the Voigt bulk and shear modulus, which are arithmetic means of the ECs with the following definition (on the basis of our C_{nn}, C_{ss} and C_{nnt}:

$$B_0 = (C_{nn} + 2 \cdot C_{nnt})/3 \tag{5.1}$$

$$G = (C_{nn} - C_{nnt} + 3 \cdot C_{nn})/5 \tag{5.2}$$

The full list of all types of cumulated ECs can be found in the appendix.

5.2 Analysis of the elastic constants

To give us a first idea about the general behavior of the elastic constants, we are going to plot the Voigt bulk and shear modulus against the lattice constants a:

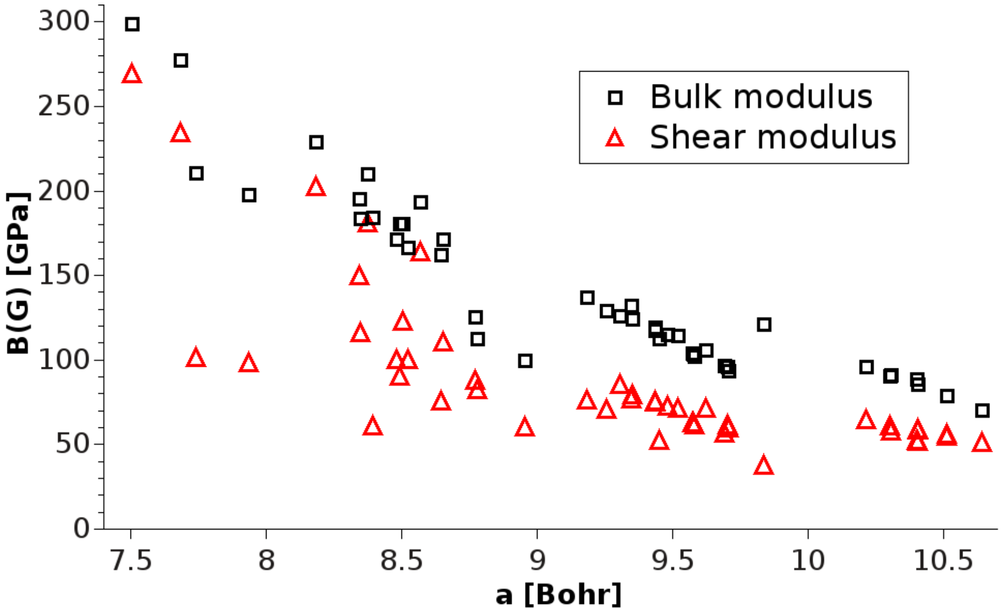

Figure 5.1: Bulk and shear moduli before correcting for a^{-4}.

We excluded carbon from this picture for better visibility.The visually most apparent correlation is that the bulk moduli are about proportional to a^{-4}. This correlation can be derived as an educated guess from theoretical considerations about bulk moduli and will usually be encountered when the atoms that are involved bind in a similar way and with similar strength, which is the case here İf we correct for the proportionality to a^{-4} by multiplying them with a^4, we get the following picture:

We see that with this correction, the B_0 and G for all configurations are within the

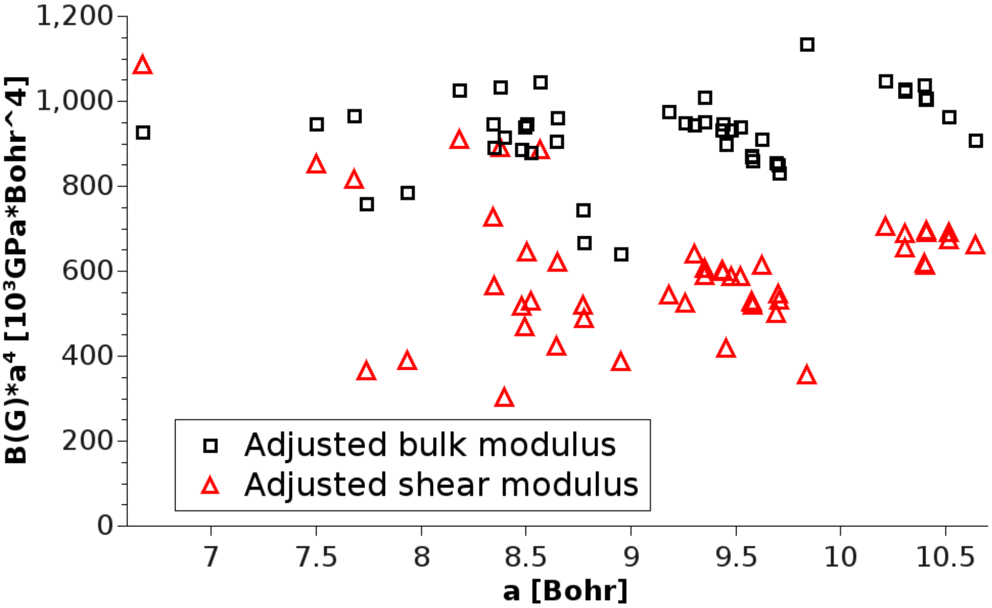

Figure 5.2: Bulk and shear moduli after correcting for a^{-4}.

same magnitude, with the corrected B$_0$ being generally around 1,000,000 GPa·Bohr^4. Meanwhile, G ranges anywhere from 360,000 to 1,100,000 GPa·Bohr^4 (if we again exclude 1122.1122, which has only 303,100 GPa·Bohr^4, since its bonds are breaking under strain). If we look closer at our data, we notice that structures with high s$_{mod}$ and broken bonds have about 10% lower bulk moduli and anywhere from 20 to 50 % lower shear moduli.

5.3 Predicting elastic constants

With these considerations made, we are going to employ a similar method to our predictions of the lattice constants. We are going to be using a Vegard-style average of the a^4-corrected ECs for pure C, Si and Ge as the basis and call it VEC_B and VEC_S, respectively:

$$VEC = x_C \cdot C_C \cdot a^4 + x_{Si} \cdot C_{Si} \cdot a^4 + x_{Ge} \cdot C_{Ge} \cdot a^4 \tag{5.3}$$

x$_i$ is again the relative content of each species. This time, we will use the respective VEC as the basis, and then add the number of broken bonds per unit cell r (list in appendix) and the number of Ge-C bonds n with a coefficient. We will then again minimize the total deviation from the calculated ECs. We do get the following formulas (in GPa·Bohr^4):

$$B_0 \cdot a^4 = VEC_B - 104,000 \cdot r \tag{5.4}$$

$$G \cdot a^4 = VEC_S - 151,000 \cdot r - 25,000 \cdot n \tag{5.5}$$

We should note that instead of the actual lattice constant a, one could also use our formula for FLC to predict B_0 and G *ab-initio*.

These formulas again improve the Vegard-style approximation massively. Even then, the formula for the bulk modulus produces an average error of about 4%, while the formula for the shear modulus produces an average error of about 14 %. This may not sound very good as such, but given that the shear modulus in our raw data (before we corrected for a^4) varied by a factor of 15, it can be deemed acceptable. The biggest outliers are again 1122.1122, where G is only about a third of the value given by the formula, and a number of carbon-germanium structures with high s_{mod} values.

We can note that the dependency on n completely vanishes for the bulk modulus (once we corrected with a^4), while it is quite sizeable for the shear modulus. It is also worth noting that the shear modulus has a much higher coefficient for the number of broken bonds. One possible explanation that we came up with is that the bulk modulus as the response to external pressure depends more on the absolute number of intact *bonds* (one broken bond would then reduce the bulk modulus by 1/16), while the shear modulus depends more on the number of *atoms* with 4 intact bonds, which can give stability to the crystal in all 3 dimensions (so one broken bond would theoretically reduce the shear modulus by 1/4, since 2 out of 8 atoms are affected). However, further research will be necessary in this regard.

Chapter 6

Conclusions and outlook

The first major conclusion we can draw from our work is that Vegard-style averages are - in our systems - not only useful to predict lattice constants, but elastic constants as well, especially for combinations of elastic constants such as the bulk and shear modulus.

The second major conclusion is that - for our systems - the most important term to fit elastic constants is a^{-4}. This behavior has been described before for group-IV elements [5].

The third and final major conclusion is that both lattice and elastic constants are - in our systems - strongly affected by the mismatch of atomic radii between carbon on one side and silicon and germanium on the other. We accounted for this with the structural parameters s (or s_{mod}) and r; we discussed their relationship in the section "Prevalence of broken bonds".

One minor conclusion to draw is that it is usually worth accounting for weak bonds such as Ge-C. The resulting corrections are not as big as the structural corrections made with s_{mod} and r in our systems, but they would be more important if the difference between the atomic radii was smaller.

One last conclusion that we have to make is that the systems we investigated constitute quite a wide range of structures, from highly stable and isotropic systems such as pure SiC to systems with high internal strain and broken bonds, where carbon switches from an sp^3 to an sp^2-hybridization. While the accuracy of our formulas for lattice constants and the bulk modulus was quite good, the average deviation of 14 % for the shear modulus formula suggests that it might be too ambitious to squeeze this range of structures into one formula.

It is reasonable to assume that these kinds of corrections that we made - with a^{-4}, with a structural parameter in the style of s, and with the number of weaker (or stronger) bonds - are useful tools to approach a new unknown system.

The fact that Vegard-style averages were quite useful in our work indicate that they would most likely be a good method if one is building a bigger cluster out of the 8-atom unit cells that we investigated. This should be especially true for lattice constants, but it might be true for bulk and shear moduli as well, as combining different unit cells should average out the more "extreme" structures.

Still, due to the all the possible effects that can occur when building a homogeneous structure out of atoms with very different atomic radii, there is a limit to *ab-initio* predictions, and one will eventually have to investigate an individual structure to determine physical properties with absolute certainty.

Bibliography

[1] *Carbon-Carbon bonds: Hybridization.* http://www.physik.fu-berlin.de/einrichtungen/ag/ag-reich/lehre/Archiv/ss2011/docs/Gina$_{P}eschel - Handout.pdf$,

[2] Birch, F.: Finite elastic strains of cubic crystals. In: *Phys. Rev.* 71 (1947), S. 809

[3] Cottenier, S.: *Density Functional Theory and the family of (L)APW-methods: a step-by-step in- troduction.* 1st. 2001 `"http://www.wien2k.at/reg_user/textbooks/DFT_and_LAPW-2_cottenier.pdf"`. – ISBN 90–807215–1–4

[4] Gulans, Andris ; Kontur, Stefan ; Meisenbichler, Christian ; Nabok, Dmitrii ; Pavone, Pasquale ; Rigamonti, Santiago ; Sagmeister, Stephan ; Werner, Ute ; Draxl, Claudia: exciting: a full-potential all-electron package implementing density-functional theory and many-body perturbation theory. In: *Journal of Physics: Condensed Matter* 26 (2014), Nr. 36, 363202. `http://stacks.iop.org/0953-8984/26/i=36/a=363202`

[5] Nolas, G.s. ; Sharp, J. ; Goldsmid, J.: *Thermoelectrics: Basic Principles and New Materials Developments.* 1st. 2001. ISSN 0933–033x

[6] P. Hohenberg, W. K.: Inhomogeneous Electron Gas. In: *Phys. Rev.* 136 (1964), Nr. B864

[7] R. Golesorkhtabar, J. Spitaler P. P. P. Pavone P. P. Pavone ; Draxl, C.: ElaStic: A tool for calculating second-order elastic constants from first principles. In: *Comp. Phys. Commun.* 184 (2013), S. 1861

[8] R. Webster, P.G. R.: *Gems: Their sources, descriptions and identification.* 1st. 2000. – ISBN 0–7506–1674–1

Chapter 7

Appendix

7.1 Used species files

Carbon species file

```
<?xml version="1.0" encoding="UTF-8" ?>
<spdb xsi:noNamespaceSchemaLocation="../../xml/species.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<sp chemicalSymbol="C" name="carbon" z="-6.00000" mass="21894.16673">
<muffinTin rmin="0.100000E-04" radius="1.2500" rinf="21.0932"
radialmeshPoints="250" />
<atomicState n="1" l="0" kappa="1" occ="2.00000" core="true" />
<atomicState n="2" l="0" kappa="1" occ="2.00000" core="false" />
<atomicState n="2" l="1" kappa="1" occ="1.00000" core="false" />
<atomicState n="2" l="1" kappa="2" occ="1.00000" core="false" />
<basis>
<default type="lapw" trialEnergy="0.1500" searchE="false" />
<custom l="0" type="apw+lo" trialEnergy="0.1500" searchE="true" />
<custom l="1" type="apw+lo" trialEnergy="0.1500" searchE="true" />
</basis> </sp> </spdb>
```

Silicon species file

```
<?xml version="1.0" encoding="UTF-8" ?>
<spdb xsi:noNamespaceSchemaLocation="../../xml/species.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<sp chemicalSymbol="Si" name="silicon" z="-14.0000" mass="51196.73454">
<muffinTin rmin="0.100000E-04" radius="1.7000" rinf="24.9760"
radialmeshPoints="300" />
<atomicState n="1" l="0" kappa="1" occ="2.00000" core="true" />
<atomicState n="2" l="0" kappa="1" occ="2.00000" core="true" />
<atomicState n="2" l="1" kappa="1" occ="2.00000" core="true" />
<atomicState n="2" l="1" kappa="2" occ="4.00000" core="true" />
<atomicState n="3" l="0" kappa="1" occ="2.00000" core="false" />
<atomicState n="3" l="1" kappa="1" occ="1.00000" core="false" />
<atomicState n="3" l="1" kappa="2" occ="1.00000" core="false" />
<basis>
<default type="lapw" trialEnergy="0.1500" searchE="false" />
<custom l="0" type="apw+lo" trialEnergy="0.1500" searchE="true" />
```

```
<custom l="1" type="apw+lo" trialEnergy="0.1500" searchE="true" />
</basis> </sp> </spdb>
```

Germanium species file

```
<?xml version="1.0" encoding="UTF-8" ?>
<spdb xsi:noNamespaceSchemaLocation="../../xml/species.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<sp chemicalSymbol="Ge" name="germanium" z="-32.0000" mass="132359.9329">
<muffinTin rmin="0.100000E-04" radius="1.7500" rinf="25.3660"
radialmeshPoints="350" />
<atomicState n="1" l="0" kappa="1" occ="2.00000" core="true" />
<atomicState n="2" l="0" kappa="1" occ="2.00000" core="true" />
<atomicState n="2" l="1" kappa="1" occ="2.00000" core="true" />
<atomicState n="2" l="1" kappa="2" occ="4.00000" core="true" />
<atomicState n="3" l="0" kappa="1" occ="2.00000" core="true" />
<atomicState n="3" l="1" kappa="1" occ="2.00000" core="true" />
<atomicState n="3" l="1" kappa="2" occ="4.00000" core="true" />
<atomicState n="3" l="2" kappa="2" occ="4.00000" core="false" />
<atomicState n="3" l="2" kappa="3" occ="6.00000" core="false" />
<atomicState n="4" l="0" kappa="1" occ="2.00000" core="false" />
<atomicState n="4" l="1" kappa="1" occ="1.00000" core="false" />
<atomicState n="4" l="1" kappa="2" occ="1.00000" core="false" />
<basis>
<default type="lapw" trialEnergy="0.1500" searchE="false" />
<custom l="0" type="apw+lo" trialEnergy="0.1500" searchE="true" />
<custom l="1" type="apw+lo" trialEnergy="0.1500" searchE="true" />
<custom l="2" type="apw+lo" trialEnergy="0.1500" searchE="true" />
<lo l="2">
<wf matchingOrder="0" trialEnergy="0.1500" searchE="true" />
<wf matchingOrder="1" trialEnergy="0.1500" searchE="true" />
<wf matchingOrder="0" trialEnergy="-1.0974" searchE="true" />
</lo> </basis> </sp> </spdb>
```

7.2 Matlab code to find configurations

```
\end{}
i=1;
j=0;
K=zeros(2,2);
```

First, we set up a list of the 6561 possible configurations.

```
for b=1:6561;
C(b)=b;
end
R(i,1)=0;
R(i,2)=1;
```

```
R(i,3)=800;
R(i,4)=11111111;
for a=2:6561
    if ne(C(a-1),6562);
        a=a-1;
        s=0;
        m=1;
        i=i+1;
        R(i,1)=a;
```

Now, we convert them into our nomenclature.

```
G(1)=floor(a./2187);
G(2)=floor(mod(a,2187)./729);
G(3)=floor(mod(a,729)./243);
G(4)=floor(mod(a,243)./81);
G(5)=floor(mod(a,81)./27);
G(6)=floor(mod(a,27)./9);
G(7)=floor(mod(a,9)./3);
G(8)=floor(mod(a,3));
for g=1:8;
    if G(g)==0;
        s=s+100;
    end
        if G(g)==1;
        s=s+10;
        end
        if G(g)==2;
        s=s+1;
        end
end
R(i,3)=s;
c=G(1).*10000000+G(2).*1000000+G(3).*100000+G(4).*10000+G(5).*1000+G(6)
.*100+G(7).*10+G(8)+11111111;
R(i,4)=c;
t=0;
for k=1:j;
    if s==K(k,1);
        K(k,2)=K(k,2)+1;
   t=1;
    end
end
if ne(t,1);
j=j+1;
K(j,1)=s;
K(j,2)=1;
end
for g=1:8;
    if G(g)==0;
        G(g)=0.0001;
    end
end
```

Now, we turn every respective configuration into a 3D-matrix resembling our crystal structure.

```
H(1,1,1)=G(1);
H(1,3,3)=G(2);
H(3,1,3)=G(3);
H(3,3,1)=G(4);
H(2,2,2)=G(5);
H(2,4,4)=G(6);
H(4,2,4)=G(7);
H(4,4,2)=G(8);
```

And lastly, we rotate, translate and mirror this 3D-matrix H in every possible way to find the number of all geometrically equivalent configurations (to our base configuration that generated H) and delete all of these configurations from our list. u, v and w represent all the diagonal directions of the crystal, x, y and z mark the possible atomic coordinates (1=0.00, 2=0.25, 3=0.50, 4=0.75) to cover translations, and the rotations and mirror symmetry are covered by permutating the powers of three at the end of each line in the six possible ways. The counter m memorizes how many geometrically equivalent configurations exist.

```
for u=[-1 1];
for v=[-1 1];
for w=[-1 1];
for x=1:4;
for y=1:4;
for z=1:4;
if ne(H(mod(x*u,4)+1,mod(y*v,4)+1,mod(z*w,4)+1).*H(mod((x+1)*u,4)+1,
mod((y+1)*v,4)+1,mod((z+1)*w,4)+1),0);
d=floor(H(mod(x*u,4)+1,mod(y*v,4)+1,mod(z*w,4)+1)*2187
+H(mod(x*u,4)+1,mod(y*v+2,4)+1,mod(z*w+2,4)+1)*729
+H(mod(x*u+2,4)+1,mod(y*v,4)+1,mod(z*w+2,4)+1)*243
+H(mod(x*u+2,4)+1,mod(y*v+2,4)+1,mod(z*w,4)+1)*81
+H(mod((x+1)*u,4)+1,mod((y+1)*v,4)+1,mod((z+1)*w,4)+1)*27
+H(mod((x+1)*u,4)+1,mod((y+3)*v,4)+1,mod((z+3)*w,4)+1)*9
+H(mod((x+3)*u,4)+1,mod((y+1)*v,4)+1,mod((z+3)*w,4)+1)*3
+H(mod((x+3)*u,4)+1,mod((y+3)*v,4)+1,mod((z+1)*w,4)+1));
if ne(d,a);
if ne(C(d),6562);
m=m+1;
end
end
C(d)=6562;
d=floor(H(mod(x*u,4)+1,mod(y*v,4)+1,mod(z*w,4)+1)*2187
+H(mod(x*u,4)+1,mod(y*v+2,4)+1,mod(z*w+2,4)+1)*729
+H(mod(x*u+2,4)+1,mod(y*v,4)+1,mod(z*w+2,4)+1)*81
+H(mod(x*u+2,4)+1,mod(y*v+2,4)+1,mod(z*w,4)+1)*243
+H(mod((x+1)*u,4)+1,mod((y+1)*v,4)+1,mod((z+1)*w,4)+1)*27
+H(mod((x+1)*u,4)+1,mod((y+3)*v,4)+1,mod((z+3)*w,4)+1)*9
+H(mod((x+3)*u,4)+1,mod((y+1)*v,4)+1,mod((z+3)*w,4)+1)*1
+H(mod((x+3)*u,4)+1,mod((y+3)*v,4)+1,mod((z+1)*w,4)+1)*3);
if ne(d,a);
if ne(C(d),6562);
m=m+1;
end
end
C(d)=6562;
d=floor(H(mod(x*u,4)+1,mod(y*v,4)+1,mod(z*w,4)+1)*2187
+H(mod(x*u,4)+1,mod(y*v+2,4)+1,mod(z*w+2,4)+1)*243
+H(mod(x*u+2,4)+1,mod(y*v,4)+1,mod(z*w+2,4)+1)*729
+H(mod(x*u+2,4)+1,mod(y*v+2,4)+1,mod(z*w,4)+1)*81
+H(mod((x+1)*u,4)+1,mod((y+1)*v,4)+1,mod((z+1)*w,4)+1)*27
+H(mod((x+1)*u,4)+1,mod((y+3)*v,4)+1,mod((z+3)*w,4)+1)*3
+H(mod((x+3)*u,4)+1,mod((y+1)*v,4)+1,mod((z+3)*w,4)+1)*9
+H(mod((x+3)*u,4)+1,mod((y+3)*v,4)+1,mod((z+1)*w,4)+1)*1);
if ne(d,a);
if ne(C(d),6562);
m=m+1;
end
end
C(d)=6562;
d=floor(H(mod(x*u,4)+1,mod(y*v,4)+1,mod(z*w,4)+1)*2187
+H(mod(x*u,4)+1,mod(y*v+2,4)+1,mod(z*w+2,4)+1)*243
+H(mod(x*u+2,4)+1,mod(y*v,4)+1,mod(z*w+2,4)+1)*81
+H(mod(x*u+2,4)+1,mod(y*v+2,4)+1,mod(z*w,4)+1)*729
+H(mod((x+1)*u,4)+1,mod((y+1)*v,4)+1,mod((z+1)*w,4)+1)*27
+H(mod((x+1)*u,4)+1,mod((y+3)*v,4)+1,mod((z+3)*w,4)+1)*3
+H(mod((x+3)*u,4)+1,mod((y+1)*v,4)+1,mod((z+3)*w,4)+1)*1
+H(mod((x+3)*u,4)+1,mod((y+3)*v,4)+1,mod((z+1)*w,4)+1)*9);
if ne(d,a);
if ne(C(d),6562);
m=m+1;
end
end
C(d)=6562;
d=floor(H(mod(x*u,4)+1,mod(y*v,4)+1,mod(z*w,4)+1)*2187
+H(mod(x*u,4)+1,mod(y*v+2,4)+1,mod(z*w+2,4)+1)*81
+H(mod(x*u+2,4)+1,mod(y*v,4)+1,mod(z*w+2,4)+1)*729
+H(mod(x*u+2,4)+1,mod(y*v+2,4)+1,mod(z*w,4)+1)*243
+H(mod((x+1)*u,4)+1,mod((y+1)*v,4)+1,mod((z+1)*w,4)+1)*27
+H(mod((x+1)*u,4)+1,mod((y+3)*v,4)+1,mod((z+3)*w,4)+1)*1
+H(mod((x+3)*u,4)+1,mod((y+1)*v,4)+1,mod((z+3)*w,4)+1)*9
+H(mod((x+3)*u,4)+1,mod((y+3)*v,4)+1,mod((z+1)*w,4)+1)*3);
if ne(d,a);
if ne(C(d),6562);
m=m+1;
end
```

```
end
C(d)=6562;
d=floor(H(mod(x*u,4)+1,mod(y*v,4)+1,mod(z*w,4)+1)*2187
+H(mod(x*u,4)+1,mod(y*v+2,4)+1,mod(z*w+2,4)+1)*81
+H(mod(x*u+2,4)+1,mod(y*v,4)+1,mod(z*w+2,4)+1)*243
+H(mod(x*u+2,4)+1,mod(y*v+2,4)+1,mod(z*w,4)+1)*729
+H(mod((x+1)*u,4)+1,mod((y+1)*v,4)+1,mod((z+1)*w,4)+1)*27
+H(mod((x+1)*u,4)+1,mod((y+3)*v,4)+1,mod((z+3)*w,4)+1)*1
+H(mod((x+3)*u,4)+1,mod((y+1)*v,4)+1,mod((z+3)*w,4)+1)*3
+H(mod((x+3)*u,4)+1,mod((y+3)*v,4)+1,mod((z+1)*w,4)+1)*9);
if ne(d,a);
if ne(C(d),6562);
m=m+1;
end
end
C(d)=6562;
end
end
end
end
end
end
end

R(i,2)=m;

end
end
```

The output is: The matrix K, which lists the 45 different chemical compositions and their respective number of geometrically inequivalent configurations, and R, which lists all 141 inequivalent configurations in our nomenclature, their chemical composition and the number of EQUIVALENT structures it is comprised of.

7.3 List of configurations, structural properties, lattice constants and bulk and shear moduli

Configuration	s_{mod}	r	n	VLC	a	B_0	
1111.1111	0	0	0	6.675	6.675	467	547
1111.1122	-0.5	0	0	7.560	7.502	299	270
1112.1112	0.5	3	0	7.560	7.741	255	102
1111.1133	-0.5	0	8	7.667	7.681	277	234
1113.1113	0.5	3	6	7.667	7.933	187	99
1111.2222	-1	0	0	8.444	8.185	228	203
1112.1222	0	0	0	8.444	8.342	193	150
1122.1212	0	0	0	8.444	8.347	183	117
1122.1122	1	(0)	0	8.444	8.395	182	61
1111.2233	-1	0	8	8.552	8.375	211	181
1112.1233	0	0	6	8.552	8.504	180	123
1113.1223	0	1	4	8.552	8.523	215	101
1122.1313	0	0	4	8.552	8.479	183	100
1123.1213	0	0	4	8.552	8.492	176	91
1122.1133	0	2	4	8.552	8.770	134	88
1123.1123	1	2	4	8.552	8.778	126	83
1111.3333	-1	0	16	8.659	8.570	193	164
1113.1333	0	0	10	8.659	8.652	169	111
1133.1313	0	0	8	8.659	8.644	165	76
1133.1133	1	2	8	8.659	8.954	107	60
1122.2222	-0.5	0	0	9.329	9.182	135	77
1222.1222	0.5	1	0	9.329	9.453	110	53
1122.2233	-0.5	0	4	9.437	9.350	125	80
1122.2323	-0.5	0	4	9.437	9.350	125	77
1123.2223	-0.5	0	2	9.437	9.304	127	86
1133.2222	-0.5	0	0	9.437	9.258	127	71
1222.1233	0.5	1	2	9.437	9.579	100	62
1223.1223	0.5	1	2	9.437	9.573	102	63
1223.1232	0.5	1	2	9.437	9.573	102	63
1122.3333	-0.5	0	8	9.544	9.520	115	71
1123.2333	-0.5	0	6	9.544	9.480	116	73
1133.2233	-0.5	0	4	9.544	9.435	119	76
1133.2323	-0.5	0	4	9.544	9.437	117	76
1223.1333	0.5	1	4	9.544	9.703	92.9	62
1233.1233	0.5	1	4	9.544	9.692	95.0	57
1233.1323	0.5	1	4	9.544	9.706	92.7	60
1133.3333	-0.5	0	8	9.652	9.657	106	72
1333.1333	0.5	1	6	9.652	9.881	82.1	38
2222.2222	0	0	0	10.214	10.214	95.7	65
2222.2233	0	0	0	10.321	10.304	89.7	61
2223.2223	0	0	0	10.321	10.305	89.8	58

Configuration	s_{mod}	r	n	VLC	a	B_0	
2222.3333	0	0	0	10.429	10.400	83.5	53
2223.2333	0	0	0	10.429	10.405	83.5	52
2233.2233	0	0	0	10.429	10.407	83.5	59
2233.2323	0	0	0	10.429	10.407	83.6	59
2222.2233	0	0	0	10.536	10.513	76.8	55
2223.2223	0	0	0	10.536	10.515	76.8	57
3333.3333	0	0	0	10.644	10.644	70.6	52

7.4 List of cumulated elastic constants

Configuration	C_{nn}	C_{ss}	C_{nnt}	C_{sst}	C_{ns}
1111.1111	1105.6	592.4	147.9	0	0
1111.1122	583.9	307.0	156.7	0	0
1112.1112	280.1	135.1	176.5	28.3	18.9
1111.1133	541.7	258.7	145.8	0	0
1113.1113	275.0	126.2	159.6	13.4	9.0
1111.2222	402.3	251.4	142.3	0	0
1112.1222	341.6	177.4	122.2	1.2	5.7
1122.1212	258.6	156.8	146.1	0	0
1122.1122	307.6	40.0	122.4	0	0
1111.2233	366.9	223.5	131.7	0	0
1112.1233	294.4	148.8	124.3	5.0	19.9
1113.1223	253.4	124.4	123.5	7.6	23.2
1122.1313	240.5	132.7	137.1	0.5	3.8
1123.1213	230.7	125.8	155.4	1.6	6.7
1122.1133	205.9	106.6	85.7	6.1	30.5
1123.1123	185.1	101.2	76.1	9.9	28.3
1111.3333	340.8	200.3	120.4	0	0
1113.1333	273.9	133.9	120.7	2.6	4.9
1133.1313	193.0	111.3	147.0	0	0
1133.1133	136.5	82.3	81.1	2.6	2.4
1122.2222	234.7	79.1	88.7	0	0
1222.1222	159.3	64.2	89.3	1.7	1.1
1122.2233	204.4	92.6	84.4	0.2	1.4
1122.2323	189.2	100.4	103.7	1.5	9.2
1123.2223	216.8	97.2	80.7	3.0	4.5
1133.2222	210.9	78.0	88.2	0	0
1222.1233	148.6	79.6	79.0	2.8	20.8
1223.1223	154.3	79.5	78.8	2.4	22.5
1223.1232	154.3	79.6	77.9	3.2	21.6
1122.3333	196.5	78.0	73.2	0	0
1123.2333	191.5	83.5	77.3	1.2	0.8
1133.2233	199.4	85.9	77.0	0	0
1133.2323	193.4	89.1	82.6	1.4	4.1
1223.1333	147.8	77.0	70.2	1.3	20.1
1233.1233	136.0	75.3	77.6	2.8	17.6
1233.1323	144.1	74.6	68.3	2.9	18.1
1133.3333	176.8	84.1	71.0	0	0
1333.1333	129.9	59.2	117.0	1.7	1.2
2222.2222	163.7	74.5	62.6	0	0
2222.2233	151.7	71.4	60.3	0	0
2223.2223	157.9	63.6	57.9	0	0

Configuration	C_{nn}	C_{ss}	C_{nnt}	C_{sst}	C_{ns}
2222.3333	122.3	71.4	72.1	0	0
2223.2333	139.3	60.4	58.9	0.1	0.1
2233.2233	143.5	69.5	56.9	0	0
2233.2323	143.2	70.3	57.0	0	0
2222.2233	140.3	61.3	48.2	0	0
2223.2223	147.0	60.2	44.9	1.4	0.9
3333.3333	120.5	61.2	45.8	0	0

7.5 Convergence tests

We did some preliminary testing and found that the more difficult parameters to converge were the elastic constants, while the lattice constants were already converged to an accuracy of 0.01 Bohr or better at much lower values for ngridk and rgkmax. Our convergence target for the elastic constants was 1 GPa. We found that pure Germanium was harder to converge than any other structure, so we will treat it separately. We picked an arbitrary ternary structure (1111.2233) for our convergence tests, and we got the following results:

Ngridk	C_{11}	C_{12}	C_{44}
2x2x2	355.5	136.3	223.3
3x3x3	371.7	134.8	224.5
4x4x4	363.3	132.6	225.1
5x5x5	364.2	132.8	224.9

Rgkmax was set to 8 for this convergence test. Based on these results, we decided to use 4x4x4 as our ngridk.

Rgkmax	C_{11}	C_{12}	C_{44}
7	369.9	131.4	225.1
8	363.3	132.6	225.1
9	362.8	132.5	225.3

Ngridk was set to 4x4x4 for this convergence test. Based on these results, we decided to use 8 as our rgkmax. We used these parameters for every configuration except for pure germanium.

For pure germanium, the convergence behavior was the following:

Ngridk	C_{11}	C_{12}	C_{44}
4x4x4	123.4	39.5	56.5
6x6x6	119.6	46.1	60.0
8x8x8	118.6	47.1	60.7

Rgkmax was set to 8.5 for this test. To be on the safe side, we decided to use 8x8x8 for our calculations.

Rgkmax	C_{11}	C_{12}	C_{44}
7.5	113.6	48.9	60.3
8.5	119.6	46.1	60.0
9.5	122.0	45.1	59.8

Ngridk was set to 6x6x6 for this convergence test. Based on these results, we decided that a value of 9 would be sufficient for rgkmax.

7.6 Glossary/Abbreviations

Chemical composition	The way the 8 atoms of a unit cell are distributed among carbon, silicon, and germanium, usually allowing for more than one **configuration**
Configuration/ Geometrically inequivalent configuration	The unique way the 8 sites of a unit cell are occupied by carbon, silicon, and germanium, with a name comprising of 8 digits
Structure/ Relaxed structure	The physical manifestation of a **configuration**, usually shifted from the ideal diamond structure due to the different atoms, same 8-digit name as the corresponding **configuration**
a	Calculated lattice constant
(L)APW	(Linearized) Augmented plane waves
$\mathbf{B}_0$	(Voigt) Bulk modulus
DFT	Density functional theory
EC	Elastic constant(s)
FLC	Formula-generated lattice constant
G	(Voigt) Shear modulus
GS	Groundstate
HK	Hohenberg-Kohn
KS	Kohn-Sham
MT	Muffin-tin
n	Number of Ge-C bonds per unit cell
s	Scalar product
s	Number of broken (“ripped”) bonds
VEC	“Vegard elastic constant”
VLC	Vegard lattice constant

Acknowledgement

I thank my supervisors Prof. Dr. Claudia Draxl and PD Dr. Pasquale Pavone.

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Berlin, den 22.8.2018

Unterschrift