Lund University

Master thesis in physics

Division of Atomic Physics, LRAP 571 (2021)

Establishing simulations of shockwaves in InSb using molecular dynamics

Author:
Erik Löfquist
Examiner:
Cord Arnold

Supervisors:
Jörgen Larsson
Eric Nilsson
Åsa Bengtsson

July 23, 2021

Simulating shockwaves atom by atom

Sending shockwaves into materials can permanently alter their structure. Uncovering the secrets of this process on the atomistic scale paves the way towards the materials of the future.

Under pressure, a lot can change. Even materials start to behave differently. At sufficiently high pressures, new materials can form. New materials are always interesting in engineering. They can both improve existing devices as well as enable completely new ones. Materials under pressure is also interesting for fundamental science. Knowing how such materials behave can give insight into for instance mineral formation.

One way to reach high pressures in materials is to send a shockwave through them. Shockwaves are waves travelling faster than the speed of sound. They arise when something moves very fast in a medium. The most familiar example of a shockwave is the sonic boom created when an airplane travels very fast in air.

Shockwaves can be sent into materials by shining laser light on them. This causes the material to heat up, melt, and rapidly expand. The expansion can happen so fast that a shockwave is created. By then sending X-rays into the material and seeing how they reflect, part of the structure can be discerned.

An experiment like this has been performed in a material called indium antimonide. After the shockwave passed, a new mysterious X-ray reflection appeared. A new material structure formed! But to know exactly what this structure is, other methods than studying X-ray reflections are required. A possible method is to simulate the shockwave in a computer, atom by atom. Such simulations are called molecular dynamics. They require many computers to work together to simulate millions of atoms simultaneously. This thesis is a molecular dynamics simulation of shockwaves in indium antimonide. It can potentially be used to explain what happens in the material as the shockwave progresses. Does for instance the mysterious X-ray reflection appear?

Indeed it does! The simulations confirm that the mysterious X-ray reflection should appear. Its origin is however yet to be determined. Close examination of the atom positions in the simulation should be able to reveal what material structure formed. If the material turns out to be useful, further molecular dynamics simulations could be used to for instance see how the material can be produced efficiently.

Abstract

Laser induced shockwaves have previously been shown to prompt structural changes in germanium. Similar experiments have now been performed in InSb, with powder diffraction patterns displaying a peak at 2.3 Å^{-1} indicating the formation of an unknown structure. In this work, large scale molecular dynamics simulations of shockwave propagation in germanium and InSb are presented. The results are in excellent agreement with published simulations in germanium, and further corroborate the experimental powder diffraction pattern in InSb. Possible structural explanations to the 2.3 Å^{-1} peak are discussed, with the formation of wurtzite InSb being unsupported. Desirable future enhancements to the model accuracy and efficiency is presented.

Acknowledgements

This work would not exist without the clear and concise guidance provided by Jörgen Larsson, to which I owe a huge thanks. The premise and goal of the master thesis felt well planned from the start, with an appropriate scope and sufficient amount of stepping stones on the way. In other words, your long experience in both research and the role as supervisor was felt and of great help to me.

I would also like to thank my co-supervisors Eric Nilsson and Åsa Bengtsson for their great patience with me. The genuine interest you showed in my progress every morning meeting was very motivating. The slight social interactions in these meetings, despite being digital, also served to decrease the negative emotions naturally present in an apartment with two stressed master students working from home.

Last but not least a great thanks to Joachim Hein and everyone else at LUNARC for all your assistance concerning the simulations on Aurora. That a complete newcomer to parallel computing and HPC-clusters was able to perform this work is proof that your training events and written guides are great and very accessible.

Contents

1. Introduction

Studying how materials change under pressure has long been of great scientific interest. An early technique to study material under static pressure that is still used today is the diamond anvil cell [1]. Extreme pressures also arise when a shockwave progresses through a material, something that has been simulated on the atomic scale since the early 80-s [2].

Laser induced shockwaves have previously been demonstrated to prompt structural changes on the nanoscale in germanium by Zhao et al.. This work included a large scale molecular dynamics simulation to validate the experimental results and study the evolution of the system as the shockwave progresses [3]. Similar experiments have now been carried out in the III-V semiconductor indium antimonide, InSb. The results indicate a feature forming in InSb not present before illumination.

The goal of this master thesis is to produce a molecular dynamics simulation replicating the experiment in InSb, in an effort to determine how InSb changes when subjected to shock. This was done in two steps: First the simulation in germanium presented by Zhao et al. was repeated, to build a reliable model and to compare the evolution in germanium to that in InSb. Then the model was modified to simulate the experiment in InSb.

The molecular dynamics simulation is built in the LAMMPS-code (Large-scale Atomic/Molecular Massively Parallel Simulator) offered by SANDIA [4]. Visualization and post-processing of results is performed in OVITO (open visualization tool) [5]. The LAMMPS code has been run in parallel on the Lund University computer cluster Aurora.

1.1. Basic Crystallography

The points around which atoms in a crystal vibrate is referred to as a lattice. Atoms in metals and semiconductors can be arranged in a periodic fashion as a lattice. A volume containing all the symmetry of the lattice, and that can be periodically repeated to recreate the entire lattice, is called a unit cell. The unit cell with smallest possible volume is called the primitive unit cell. Many materials, such as aluminium, germanium, and bulk InSb have cubic unit cells. Aluminium atoms are positioned in the corners of the unit cell and on the center of the cube faces, forming a face centered cubic lattice. In germanium every atom bonds to four neighbours in a regular tetrahedron shape, forming a diamond cubic lattice. The indium and antimony atoms in InSb bond analogous to a diamond lattice, with each atom bonding to four

atoms of the different type. This is known as a zincblende lattice. It can be viewed as two interwoven face centered cubic lattices [6]. While bulk InSb has a zincblende structure, InSb nanowires can also be grown in a hexagonal structure called wurtzite [7].

The three vectors constructing the unit cell are used for denoting directions and planes in the lattice, conventionally with Miller indices h,k,l. They form a Cartesian coordinate system for cubic lattices. Directions are enclosed as [hkl] and planes as (hkl). The set of all by symmetry equivalent directions are written as <hkl >, and similarly for planes as {hkl}. Negative directions are denoted with an overbar [6].

A lattice can be further characterized using the radial pair distribution function (RDF) g. It is a probability distribution that describes the probability of finding another particle at a radial distance r from an atom. If all atom positions are known it can be calculated by collecting the distances between all pairs of atoms in the crystal in a histogram [8].

1.2. X-ray Diffraction

X-ray diffraction, XRD, is a technique that can be used to identify structures on the atomic scale. When X-rays encounter atoms in a lattice plane (hkl) they partially reflect. While each partial reflection is small, if all reflections from consecutive parallel planes add up constructively nearly all X-rays reflect from the crystal. This occurs when the Bragg condition is satisfied, at

$$\frac{1}{d} = \frac{2\sin\theta}{\lambda} \tag{1}$$

where d is the distance between adjacent planes, θ the angle of the incident wave with respect to the lattice plane, and λ the wavelength of the X-rays. Sending X-rays into a crystal, varying angles and studying the reflections can thus be used to determine the characteristic distances of the crystal, and from this deduce its structure [9].

In so called powder diffraction, one assumes that the material has been grained to a powder consisting of many small crystals that locally exhibit the same crystal structure but with random orientations. When sending a beam into such a powder there will always be some grains that are orientated such that Bragg's law can be satisfied. The reflected X-rays for a particular spacing d will form a cone (again due to the random crystal orientations) of half-apex angle 2θ with respect to the incident beam. In this manner, an intensity pattern as a function of reflected angle, featuring all XRD-peaks

resolvable with the used wavelength λ, can be measured in one shot without varying the incidence angle [9].

The diffracted angle θ in Bragg's law (equation (1)) depends on the wavelength λ of the X-ray source. To make measurements between different X-ray sources comparable, the XRD pattern can instead be expressed as a function of the scattering vector q. It can be introduced as [8]:

$$q = \frac{4\pi \sin\theta}{\lambda} \tag{2}$$

The scattering vector can be calculated given the diffraction angle and the wavelength of the X-ray source. Combining equation (1) and (2) we find the relation between the scattering vector and the distances between planes as:

$$q = \frac{2\pi}{d} \tag{3}$$

From equation (3) it is evident that the value of the scattering vector is only related to the lattice and not the wavelength of the X-ray source.

The scattered intensity is also referred to as the structure factor S [8]. (In some literature, the structure factor denoted F instead refers to the scattered amplitude [9]. In this case S is the absolute square of F). The intensity as a function of scattering vector q, $S(q)$, will be referred to as the XRD-pattern. The XRD-pattern is linked to the RDF through a sine Fourier transform [8]:

$$S(q) = 1 + \frac{1}{q}\int_0^\infty g(r)\sin(qr)dr \tag{4}$$

There are many different definitions of the RDF differing in normalizations and additive constants [10]. However, due to the linearity of the Fourier transform, the different definitions of the RDF will always yield XRD peaks at the same scattering vectors q.

Calculating the RDF for large radii is computationally expensive for millions of atoms. Truncating the RDF at a cut-off radius r_c, and multiplying by a window function $w(r)$ to mitigate the Fourier components of the sharp truncation, equation (4) can be approximated as:

$$S(q, r_c) \approx 1 + \frac{1}{q}\int_0^{r_c} w(r)g(r)\sin(qr)dr \tag{5}$$

1.3. Molecular Dynamics

A useful introduction to the key concepts of molecular dynamics is given by Gilead B. Tadmor and Ronald E. Miller in "*Modeling materials : continuum, atomistic and multiscale techniques*" [11]. Molecular dynamics, MD, aims to simulate materials by modeling the movements of individual atoms over time. The atoms are treated classically as point particles. Their interactions are governed by a potential V, implicitly defined through:

$$F = -\frac{\partial V}{\partial r} \tag{6}$$

where F is the force on each atom and r denotes the distance between the atoms. In MD the potential is assumed to be known, expressed in such a way that one can calculate the forces on each atom through equation (6). Summing all forces on the individual atoms, their motions can be extracted by integrating Newtons equations in time. This is repeated for discrete times separated by a timestep on the order of 1 fs, which is typically enough to resolve the atomistic vibrations with highest frequency [11].

There are many different ways to express the potential function. A historically important potential commonly used in MD is the Lennard-Jones potential. It is given by:

$$V(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \tag{7}$$

where V is the potential, and r the radial distance from the nucleus. The parameters ϵ and σ describe the depth of the created potential well, and the radius at which the potential is equal to zero, respectively. By varying these parameters different molecules or atoms can be modeled. The behaviour of equation (7) is illustrated in figure 1. The strong repulsion for short radii models the Pauli exclusion principle, that forces electrons into higher energy states as atoms get close. The attractive tail governing the long range interactions model the London dispersion (i.e. the attractive force due to fluctuations in the charge field) [12].

The Lennard-Jones potential is a so called two body potential, as it only describes pair-wise interactions between all particles. The potentials used in this work, embedded atom potential [13] for aluminium, Tersoff potential [14] for germanium, and Vashishta potential [15] for InSb, are all three body potentials. The inclusion of higher order terms are used to model asymmetry

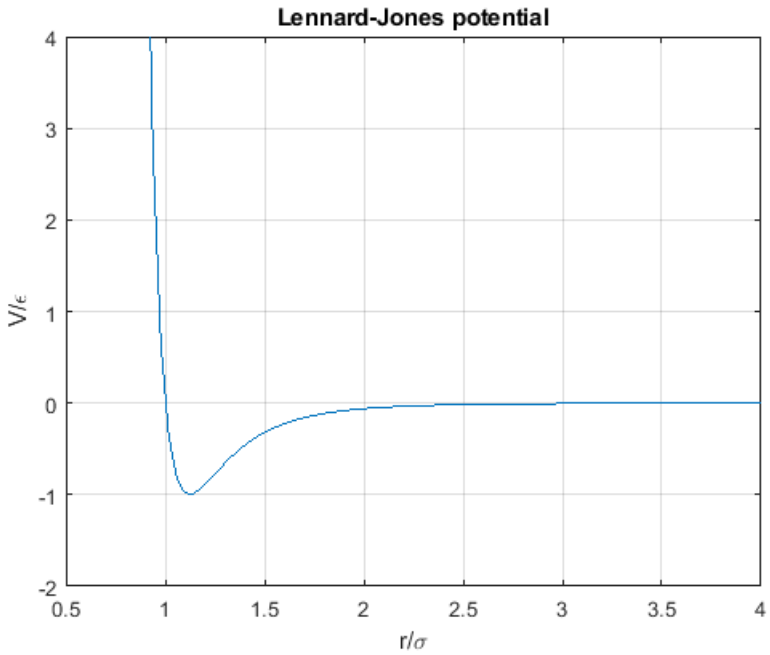

Figure 1: The Lennard-Jones potential energy plotted as a function of radius.

of the wavefunction in bonded interactions [12]. In the Tersoff [14] and Vashishta potential [15], this is expressed by the potential energy depending on the angle between bonds. Apart from this bond-angle energy, and the forces modeled by the Lennard-Jones potential, the Vahsistha potential also models screened Coulombic and screened charge-dipole interactions [15].

An isolated system where the number of particles N, volume V, and energy E is conserved is called an nve-ensemble or micro-canonical ensemble. One can also simulate a system in contact with a heat bath of temperature T using an nvt-ensemble, or canonical ensemble. In an nvt-ensemble the temperature of the system is controlled by a so called thermostat, that rescales particle velocities between timesteps. Analogously a barostat rescales atom positions to achieve a target pressure. Temperature is measured through the time averaged vibrational kinetic energy of the system [11].

Any new atom positions and velocities thus depend both on the ensemble used, its thermostat and barostat, and the integration of Newtons equations in time. The acquiring of new positions and velocities can be shortened as "nve-integration", from which it is implied that it is Newtons equations that are integrated in time with a thermostat and barostat appropriate for the nve-ensemble [16]. Similar terminology is used for the nvt-ensemble [17].

When setting up an MD simulation of a crystal, initial positions and velocities must be defined. The initial positions are typically the perfect lattice sites, while the initial velocities can be randomized to a distribution corresponding to a target temperature. If such a system is integrated as an nve-ensemble, the system temperature will drop below the target temperature over time. This is due to energy conservation: As the perfect lattice sites correspond to potential energy minima, the kinetic energy and thus the temperature will decrease when displacements occur and the potential energy increases. The desired target temperature can be reached by first integrating the system as an nvt-ensemble, a process known as equilibration, before running the nve-integration [11].

To avoid modelling boundary effects and mimic a bulk material, many MD simulations employ periodic boundary conditions. These work by replicating the simulation domain and all its particles by translation infinitely at the boundary. A 2D example of one such replication is shown in figure 2. Atoms near the simulation box boundary will interact with atoms on the opposite side of the box. In figure 2 the leftmost blue triangle would feel a force to the left from the red diamond in its simulation box, as well as a force to the right from the diamond in the other box. Atoms passing through the boundary

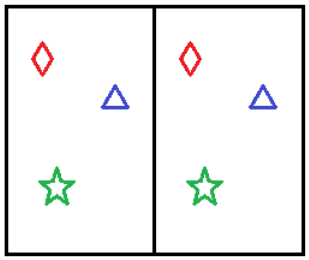

Figure 2: Schematic sketch of a rectangular simulation box replicated once at a periodic boundary. The particles, illustrated by coloured shapes, have exactly the same position and velocities in both images.

will end up on the other side of the simulation box, and atoms positioned exactly at the boundary can be viewed as being situated at both sides of the box at once. One can show that solving the MD problem for the finite number of atoms confined in the simulation domain also solves the problem for the infinite replicas [11].

As atoms on a periodic boundary can be viewed as being situated on both ends of the simulation domain, these two planes must be identical to create a perfect lattice. The simplest case is a simulation consisting of a single unit cell. As the unit cell by definition can create the lattice by replication and translation alone, periodic boundaries can be applied to all its faces. More generally, a box aligned with the faces of an integer number of unit cells will create a periodic system. Note that the simulation box length is therefore related to the periodicity of the lattice.

All simulation boxes in this work are rectangular cuboids, and both germanium and bulk InSb have cubic unit cells. Orienting the simulation box in [100], [010], and [001] can therefore create a periodic system. However, when aligning the simulation box in [111] the faces of the unit cells no longer coincide with the faces of the simulation box. It is then non-trivial to find how to orientate the simulation box orthogonal to [111], and how to choose the box lengths in order to create a periodic lattice (if a solution exists).

To describe the deformation of a body one uses strain and stress. Strain is the change in length of a body when loaded, divided by its initial length when unloaded. In MD it can be directly calculated from the atom positions. Stress σ is the applied force per unit area on a body. The considered area is the cross section perpendicular to the applied load. In MD it is desirable to

calculate the stresses not only on a body, but also on each individual atom. Per-atom stresses are however not straightforward to calculate, as the volume belonging to an atom (and thus the relevant area) in a deformed crystal is ill-defined. Instead of a true per-atom stress tensor, LAMMPS instead uses a stress times volume formulation [18].

An approximation to the volume belonging to one atom is the Voronoi-volume. It is the volume consisting of all space that is closest to a particular atom compared to all other atoms. This can be used to obtain an approximate stress tensor [18]. From the elements σ_{ij}, $i, j = 1, 2, 3$, of the stress tensor one can calculate the so called von Mises stress σ_v. It is a commonly used deformation criterion in engineering that represents the entire stress tensor as a single scalar through: [19]

$$\sigma_v^2 = \frac{1}{2}((\sigma_{11} - \sigma_{22})^2 + (\sigma_{22} - \sigma_{33})^2 + (\sigma_{33} - \sigma_{11})^2) + 3(\sigma_{12}^2 + \sigma_{23}^2 + \sigma_{31}^2) \quad (8)$$

As the MD problem for many potential types consists of evaluating pairwise forces between N atoms, the computational cost to take one timestep grows proportional to N^2. This growth is too fast to model millions of particles. By splitting the force interaction into short range interactions (determined explicitly for all atom pairs in real space) and long range interactions (determined collectively in reciprocal space) at a certain cut-off radius r_c, the computational cost can be reduced to be proportional to $N\log N$ [20].

In LAMMPS the cut-off radius is specified by the potential. Some MD implementations only consider the short range interactions. Incorporating neighbour lists, that track which atoms are likely to interact on the next timestep, are used to reduce the number of computations. This avoids calculating all atom pair distances in the simulation each timestep. Another approach is to use binning, where the simulation domain is subdivided into smaller regions. Atoms then only need to check for interactions in their bin, and the adjacent bins within the cut-off radius [11].

LAMMPS uses both neighbour lists and binning paired with parellelization, where each bin is handled by a separate processor. The MD problem is not perfectly parallelizable as the processors must communicate atom positions to evaluate forces, update neighbour lists, and account for atoms moving between adjacent bins.

1.4. Experiment in InSb

The, as of July 23, 2021, unpublished experiment was conducted at the X-ray pump-probe instrument at the LCLS (Linac Coherent Light Source)

facility. This instrument combines an ultrafast laser to pump a sample with light, with a free-electron laser (FEL) that probes the sample with X-rays [21]. The used sample consisted of a crystal of InSb with a 300 nm thick layer of aluminium deposited on the (111) surface. The aluminium was illuminated with a 0.5 ps laser pulse at 800 nm, with a fluence of 400 mJ/cm^2. The laser pulse was absorbed in the aluminium layer, causing it to rapidly melt. As the density of liquid aluminium is lower than in solid aluminium, this led to a violent expansion of the material. The expanding aluminium acted as a piston compressing the InSb, sending a shockwave into the material.

The shocked sample was probed with X-rays from the FEL, also incident from the aluminium surface. The experiment was repeated around 100 000 times, varying the delay between the laser pulse and the X-rays (from around 75 ps to several μs) as well as the laser power and beam incidence angles.

One of the measured XRD patterns are shown in figure 3. The XRD before, during and after the laser pulse was sent is shown in blue, yellow, and red, respectively. The three intense peaks seen before the laser pulse was sent corresponds to crystalline aluminium. These disappear as the aluminium melt. No InSb peaks are observed before the laser pulse is sent as the crystal had been rotated such that no reflections occurred.

After the shockwave had propagated through the InSb, three intense peaks appeared that correspond to the {111}, {220}, and {311} reflections of crystalline InSb. These were now visible due to the shockwave changing the orientation of parts of the material. However, at 2.3 Å^{-1} a distinct peak not present in In, Sb or zincblende InSb can also be seen.

Determining the structural origin of the XRD of the shocked material motivates this molecular dynamics study. An in-going hypothesis is that regimes of wurtzite InSb might have formed, as this material displays an XRD peak at 2.3 Å^{-1}.

2. Method

2.1. The LAMMPS-input script

The used LAMMPS input script for shock compression is an extension of a script presented by Guerrero-Miramontes [22]. The input script in its entirety, along with a beginner's guide to LAMMPS, OVITO, and utilizing computer clusters, can be found in appendix E.3, A, B, and C, respectively.

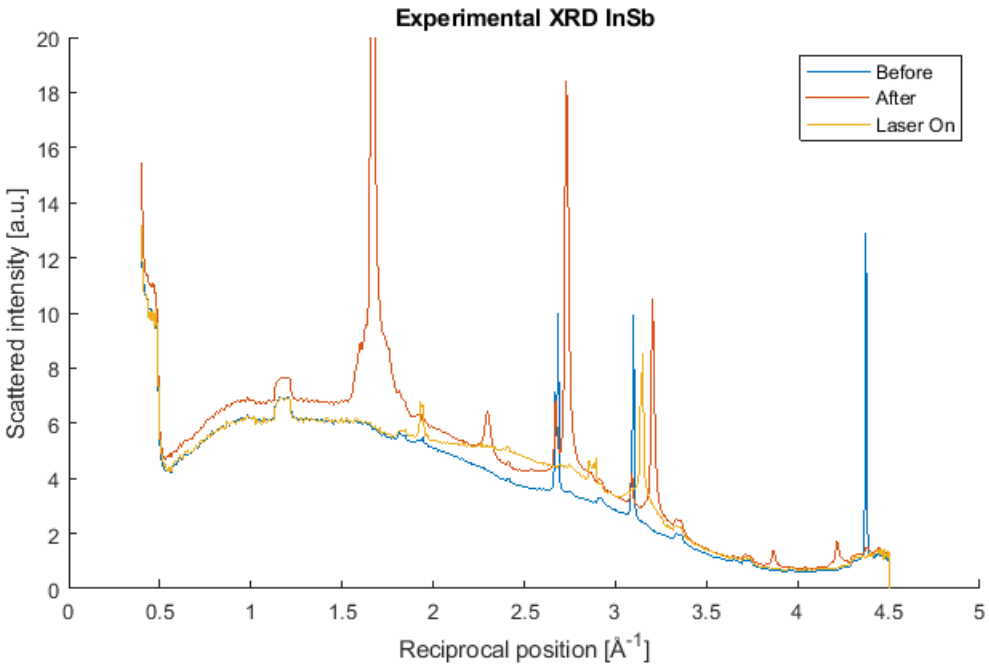

Figure 3: Experimentally obtained XRD pattern in InSb. Blue curve is before the laser pulse was sent, yellow curve during the melting, and red curve afterwards.

2.1.1. Modeling shock in LAMMPS

A simulation box of desired dimensions is created aligned with a Cartesian coordinate system. Periodic boundary conditions are applied to all sides of the simulation box. The simulation box is filled with atoms on a lattice of desired orientation. The atoms are randomly assigned initial velocities according to the Maxwell-Boltzmann distribution at a user defined temperature.

Temperature and zero internal pressure is achieved by integration as an nvt-ensemble. The used thermostat and barostat is a Noose-Hover formalism modified to dampen temperature and pressure oscillations [17].

After equilibration, the periodic boundary condition along the compression direction is lifted and the system integration is changed to an nve-ensemble. Atoms in a bottom layer of four unit cells in the compression direction are selected to act as a piston. The piston is accelerated to a user defined piston velocity towards the material, sending a shockwave into the crystal.

In the original script by Guerrrero-Miramontes the piston is instantly set to the desired piston velocity [22]. This script has been extended to include a linear acceleration of the piston up to the target velocity over a user defined time. This matches the methodology used by Zhao et al. studying shock induced dislocations in germanium with an acceleration time of 1 ps [3], and Hahn et al. studying similar dislocations in silicon with an acceleration time of 2 ps [23]. The acceleration time used in this work is 1 ps to match the work by Zhao et al.

In order to also study not only the compression but also the relaxation of the material, the script in this work also includes a possibility for the user to linearly decelerate the piston to rest over a user defined time. This stops inducing shockwaves and allows the material to relax. The piston is kept stationary after the linear deceleration.

When running the script on multiple processors the simulation box is subdivided on a rectilinear grid, assigning each processor to update the atoms within a specific volume. This mapping is done such that the surface area between adjacent processors is minimized. For a homogeneous distribution of particles, this also gives roughly equal amount of atoms to each processor. Compressing the simulation box leads to an in-homogeneous distribution of particles in space and thus uneven computational load between adjacent processors. To mitigate this the script includes re-balancing after user defined

time intervals as detailed in appendix E.3. At each re-balance the partitioning between processors in the compression direction is changed in such a way that the atoms are as evenly distributed on the processors as possible.

A variant of the script using non-periodic boundary conditions during both the equilibration and the shock sequence has also been studied. These simulations used a Berendsen thermostat [24] during equilibration. This was done as the Nose-Hoover thermostat in LAMMPS requires periodic boundaries [17].

2.1.2. Generated outputs

Apart from the default command window output and LAMMPS log-file where the used initial conditions and the total walltime is written, the input script also writes several additional files. System characteristics during the equilibration is written to "initialConditions.out". These are the current timestep, the pressure times volume components along all axes, simulation box dimensions, temperature, and the total energy of the system. During the run the same characteristics is written to "run.${temperature}K-.averagedQuantities.out", along with the center of mass in the compression direction.

Atom-wise quantities are dumped as cfg-files "dump._${currentTimestep}-.cfg" that can be read by OVITO. They include each atoms unique id, mass, type, coordinates, potential energy, kinetic energy, stress times volume tensor, Voronoi volume and coordination number.

Along the compression direction several per-atom quantities are averaged and output as a function of position. This is done by dividing the simulation domain in slabs normal to the compression direction, referred to as "chunks" in LAMMPS. A per-atom quantity is averaged over all atoms in a chunk to get a value for that chunk. These values are then further sampled and averaged in time before the output is written to a file. This reduces both the output data size as well as the total computation time. The output quantities are mass density, pressure times volume, temperature, and velocity component in the compression direction (output both for the bulk material and the piston separately). They are written to "denz.profile", "pressure.profile", "temp.profile", "velocityZcomp.profile", and "velocityZpiston.profile" respectively.

2.2. Post-processing in OVITO

In OVITO the shockwave is visualized by colouring atoms with the dumped per-atom quantities. The images in this work uses colours describing either the coordination number or the von Mises stress. The latter is acquired by taking each element in the dumped stress times volume tensor, dividing by the Voronoi volume, and using the elements in this stress tensor in equation (8) to get the von Mises stress.

To examine the structure of the compressed material the "Identify Diamond Structure" algorithm implemented in OVITO was used. It runs common neighbour analysis extended to the second neighbours of atoms to determine if an atom is in a cubic diamond or hexagonal diamond environment [25]. For the algorithm to correctly identify the structure of material that is uniaxially compressed, an affine transformation stretching the material to its initial length is first applied. After performing the structure identification, an inverse transformation is applied to return the material to its actual state. This methodology follows the one presented by Zhao et al. [3]. An concrete example of how to perform this in OVITO is given in appendix B.2.

The radial pair distribution function was calculated from definition, by measuring the radial distance between atoms within a radial cutoff and binning the distances into a histogram. (Note that this creates a normalization that depends on the number of binned atoms). Sine transformation of the radial pair distribution function according to equation (5) was used to get powder diffraction patterns. A sine-window was used as the window function. A short MATLAB-script implementing this is found in appendix E.1.

2.3. Performed production runs

All simulations were equilibrated to a temperature of 300 K and zero pressure. The binned outputs were performed using 3 Å thick bins, writing to the file every 100 timesteps. Chunk-averaging was employed five times separated by 10 timesteps for each data point. Shock simulations that did not study the relaxation of the material included no piston deceleration. All shock simulations in InSb used an initial lattice of zincblende structure.

The initial lattice constants were chosen as 5.658 Å for germanium and 6.479 Å for InSb. The de-facto lattice constants arise after the equilibration and are thus determined by the used potential. Germanium was simulated with a mass of 72.64 u, indium with 114.8 u, and antimony with 121.8 u [26].

A Tersoff potential was used for simulating germanium, with parameters published by Tersoff [27]. A 40x40x60 nm crystal was compressed along

[001] for piston speeds of 1.1, 1.2, and 1.4 km/s. The timestep was 1 fs, equilibration time 10 ps, and the shock sequence (measured from the start of the piston acceleration) was 8.75 ps.

A Germanium crystal with identical dimensions was also compressed along [111]. This run used a piston speed of 1.2 km/s, timestep of 2 fs, an equilibration time of 8 ps, and a 8.75 ps shock sequence.

InSb was simulated using a Vashishta potential with parameters published by Rino et al. [28]. A 40x40x60 nm crystal was compressed along [001] using a piston speed of 1.2 km/s, timestep of 1 fs, equilibration time 10 ps, and 8.75 ps shock sequence.

40x40x60 nm InSb crystals has been compressed along [111] for piston speeds of 0.9, 0.95, 1.0, 1.1, 1.2 and 1.3 km/s. These runs used a 2 fs timestep, 8 ps equilibration, and a 12 ps shock sequence, except the run with piston speed 1.2 km/s that used a 1 fs timestep, 10 ps equilibration, and a 10 ps shock sequence. A run with piston speed 1.1 km/s, timestep 1 fs, equilibration 10 ps and shock sequence of 12 ps was also performed. Runs in identical crystals with non-periodic boundary conditions have been performed over the same times for piston speeds of 0.4, 0.6, 1.0, 1.1, 1.15, 1.2, 1.4, 1.6, and 2.0 km/s.

Relaxation of a 40x40x60 nm InSb crystal compressed along [111] was also studied for a total simulation time of 14 ps. The piston was accelerated to 1.2 km/s over 1 ps, held at this velocity for 5 ps, decelerated over 2 ps, and then kept stationary the remaining time. Equilibration time was 10 ps and the timestep 2 fs. This simulation was also extended to a longer timescale of 50 ps in a 40x40x200 nm crystal with identical piston behaviour.

In addition to shock simulations, XRD patterns of wurtzite InSb and face centered cubic aluminium crystals were also simulated. Both simulations used a 20x20x20 nm crystal equilibrated to a temperature of 300 K and zero external pressure, using an equilibration time of 10 ps and a timestep of 2 fs. Wurtzite InSb was simulated using the same potential as the zincblende, while the aluminium used an embedded atom model potential with parameters published by Mishin et al. [29].

The RDF was calculated for the entire crystals for all simulations except those with compression along [111]. For these simulations, only the central 20x20 nm in (111) were used in order to avoid sampling the boundaries. The RDF was truncated at 80 Å in the runs with compression along [001], 70 Å in the wurtzite InSb crstal, and 50 Å for all other simulations.

3. Results

The walltimes and node configuration for some typical production runs can be found in appendix D.

3.1. Germanium

3.1.1. Compression along [001]

Germanium was first compressed in [001] to compare the results to those of Zhao et al. [3]. To verify that a shockwave emerges and behaves as expected, and to see what features in shocked germanium arise due to directional amorphization, a piston speed below the reported amorphization threshold was first examined. With a piston speed of 1.1 km/s, all atoms remain in a diamond cubic structure strained along [001]. The atom velocity component along the compression direction (found in the file "velocityZ-comp.profile") are illustrated as a function of position in figure 4. All atoms the shockfront has reached are accelerated to and maintained at the piston velocity of 1.1 km/s. The shock speed, estimated from where the particle velocities have reached 0.2 km/s, is on average 5.41 km/s.

The XRD (calculated by the code in appendix E.1) of the unshocked material compared to the shocked material 8.75 ps later is shown in figure 5. Peak positions in the unshocked material agree with that of a diamond cubic structure. When compressing the intensity of most original peaks are lowered, while new peaks emerge adjacent to the old ones. This is likely due to strain shifting of the original peaks. No X-rays are reflected between the peaks, indicating that the material is still crystalline.

Next the piston speed was increased to 1.2 km/s, the amorphization threshold reported by Zhao et al. [3]. This claim is verified, directional amorphization occurs in bands in {111}. The system is illustrated at time 8.75 ps in figure 6 where atoms in a diamond cubic structure are coloured by coordination number within a cut-off radius of 10 Å and the amorphous atoms marked in red. Atoms that are second neighbours to diamond cubic atoms are coloured gray. The piston is at the bottom. The shockwave can be seen propagating upwards in the figure as the coordination number increases abruptly at the shockfront. The coordination number decreases slightly between the amorphous bands and the shockfront. The bands are shown with all diamond cubic atoms removed in figure 7. (A detailed description of how these figure were created is found in appendix B.2).

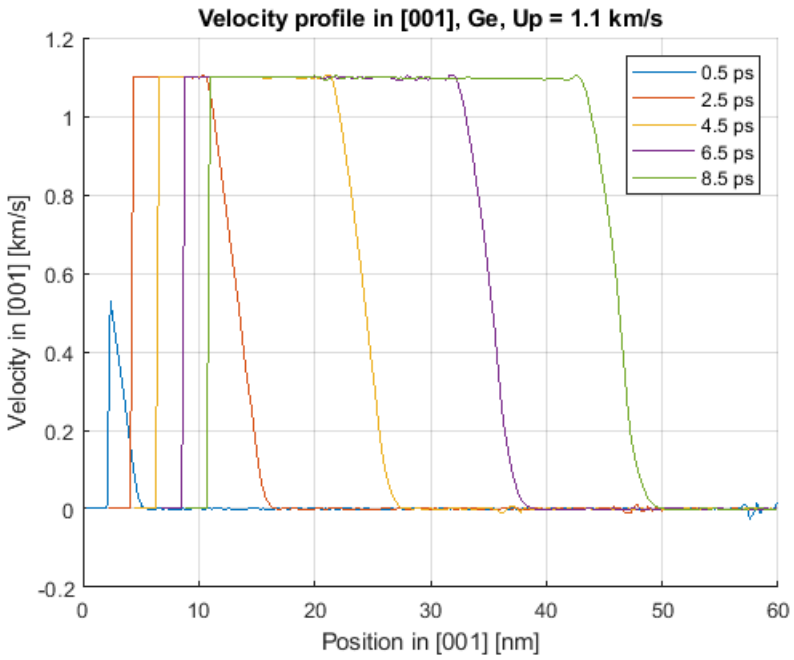

Figure 4: Velocity in germanium along [001] as a function of position for five different times. Piston speed is 1.1 km/s.

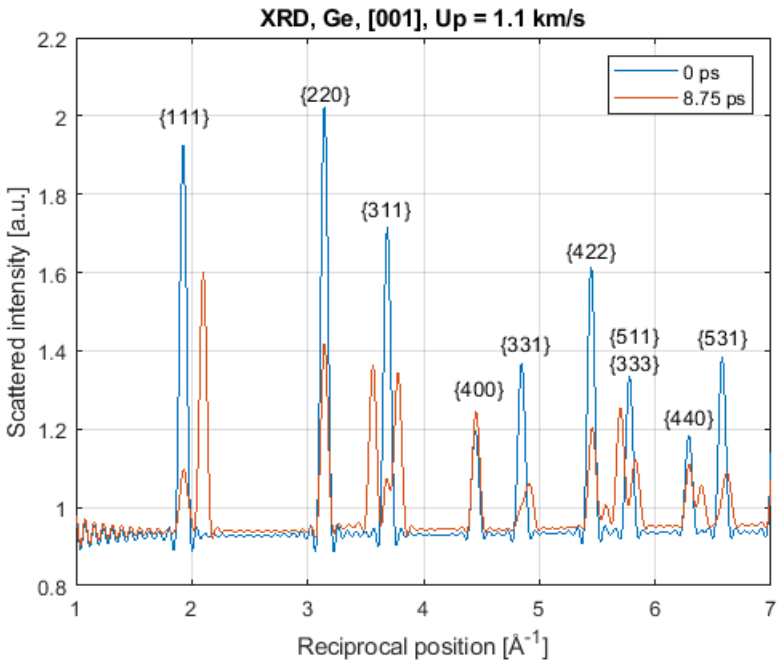

Figure 5: XRD pattern of germanium compressed in [001] with piston velocity 1.1 km/s. Crystalline germanium is in blue, shocked in red.

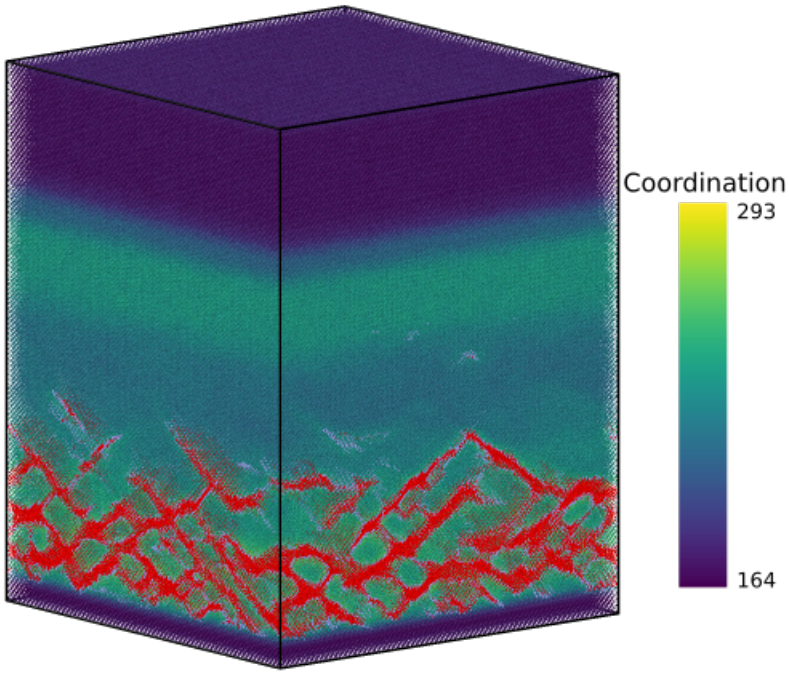

Figure 6: Germanium after 8.75 ps of compression in [001] coloured by coordination number within 10 Å. Atoms in an amorphous environment are coloured red and their neighbours grey.

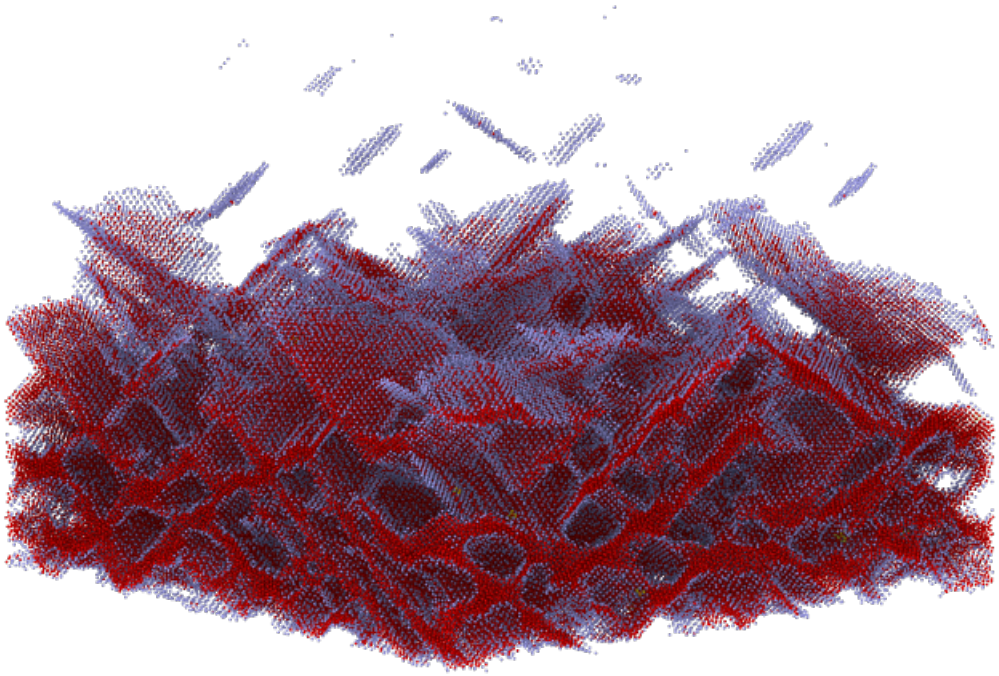

Figure 7: The amorphous bands in figure 6 without the surrounding atoms. The bands roughly align with {111}.

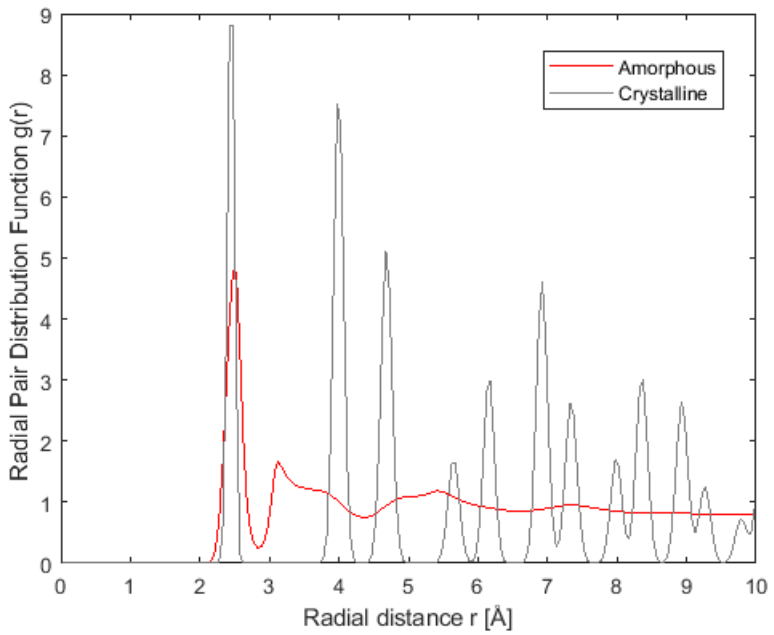

Figure 8: RDF for the germanium atoms part of the amourphous bands. Grey curve is before compression, and red curve is after 8.75 ps compression in [001].

The radial pair distribution function is shown in figure 8 for the atoms in the amorphous bands, at time 0 ps when they are in a diamond cubic environment and at time 8.75 ps when they are part of the amorphous bands. The crystalline peaks are nearly completely smeared out above a radial distance of 3 Å.

Studying the velocity profile in figure 9, atoms are still initially accelerated to the piston velocity at the shockfront. However, a reduction in velocity occurs between the atoms close to the piston and those at the shockfront. It is further reduced over time, and extends over a larger position range as the shockwave progresses. This implies that the formation of the amorphous bands slow down the surrounding atoms. A possible explanation is that forming the amorphous bands requires some potential energy, which is taken from the kinetic energy of the atoms. The velocity reduction is likely also the cause of the reduction in coordination number between the shock front and the amorphous bands in figure 6, which is also seen in the figure presented by Zhao et al. The shock speed, estimated from where the particle velocities

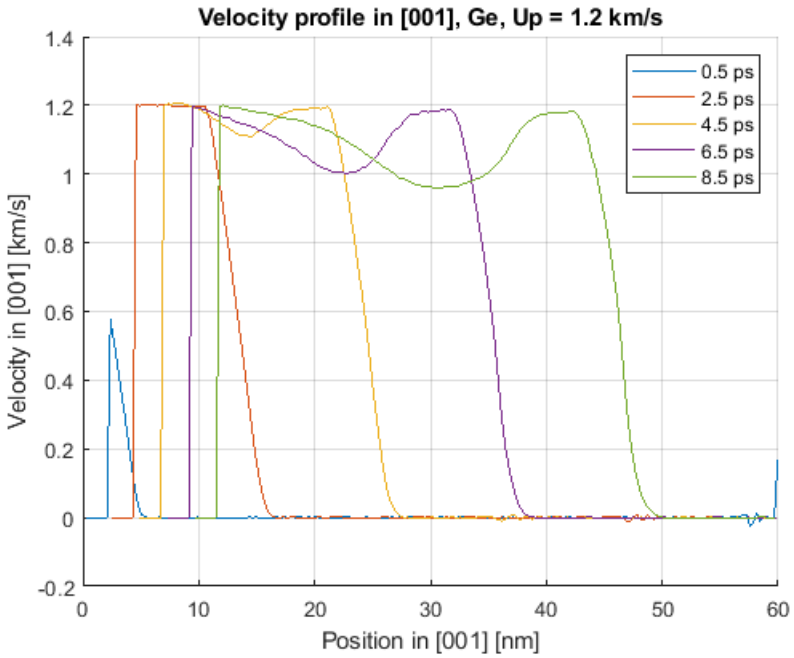

Figure 9: Velocity in germanium along [001] as a function of position for five different times. Piston speed is 1.2 km/s.

have reached 0.2 km/s, is on average 5.42 km/s.

The XRD of the unshocked material compared to the shocked material 8.75 ps later is shown in figure 10. As for the XRD with piston speed 1.1 km/s in figure 5, the original crystalline peaks are lowered as adjacent strain shifted peaks emerge. However for a piston speed of 1.2 km/s, some X-rays are scattered for all positions above 2 Å^{-1}. This further suggests that the formed bands are amorphous. The uniformly scattered X-rays in the amorphous bands also lead to a lower intensity of the strain shifted peaks due to energy conservation.

The piston speed was increased in order to see how the amorphization is affected, and if the trends seen between the piston speeds of 1.1 and 1.2 km/s hold for larger piston speeds. For a piston speed of 1.4 km/s, even more amorphous bands in {111} form. Near the piston they overlap to form a complete layer of amorphized atoms.

In figure 11 which shows the velocity profile, the velocity reduction seen for a piston speed of 1.2 km/s in figure 9 is further amplified. This again

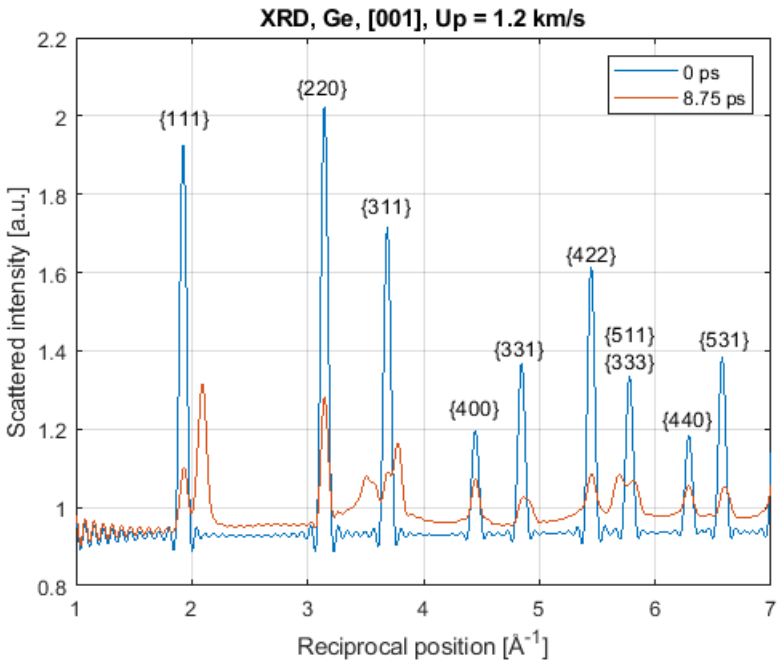

Figure 10: XRD pattern of germanium compressed in [001] with piston velocity 1.2 km/s. Crystalline germanium is in blue, shocked in red.

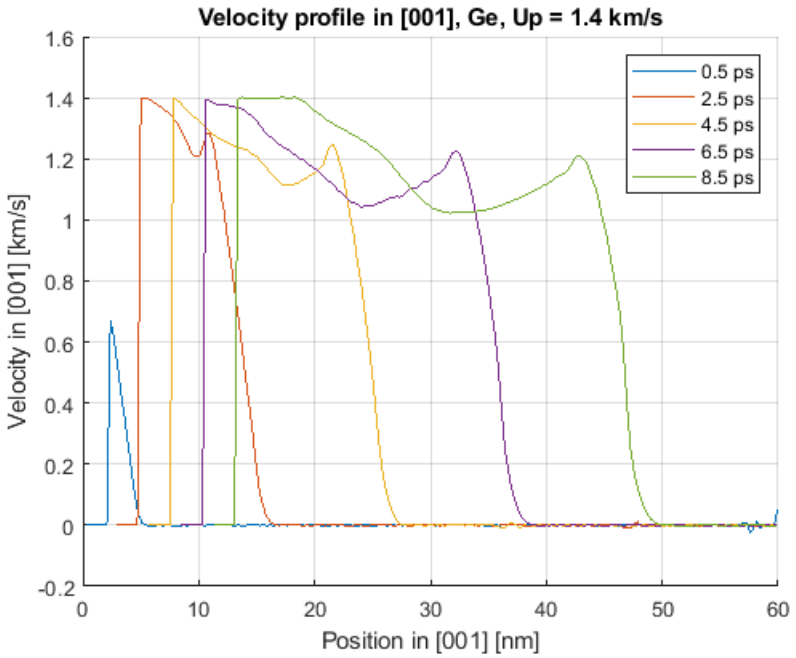

Figure 11: Velocity in germanium along [001] as a function of position for five different times. Piston speed is 1.4 km/s.

suggests that the velocity reduction is coupled to the amorphization process. Atoms near the shockfront no longer reach the piston speed of 1.4 km/s. The shock speed measured at 0.2 km/s is on average 5.42 km/s, similar to the speeds of 5.41 and 5.42 km/s obtained for piston speeds of 1.1 and 1.2 km/s. It is also close to the shockspeed of 5.2 km/s reported by Zhao et al. [3].

Overall the results when compressing germanium along [001] are in excellent agreement with those reported by Zhao et al. [3]. Directional amorphization in {111} occurs above the reported critical piston velocity of 1.2 km/s, with the bands created and identified with identical methodology. The system coloured by coordination number in figure 6 as well as the amorphous bands in figure 7 and the radial pair distribution function in figure 8 all compare well to the figures presented by Zhao et al. (The exact position of the bands are not the same, as the locations vary randomly between successive runs). The similarities between the obtained results and those presented by Zhao et al. suggests that their implementation of the MD-simulation resembles the one presented by Guerrero [22].

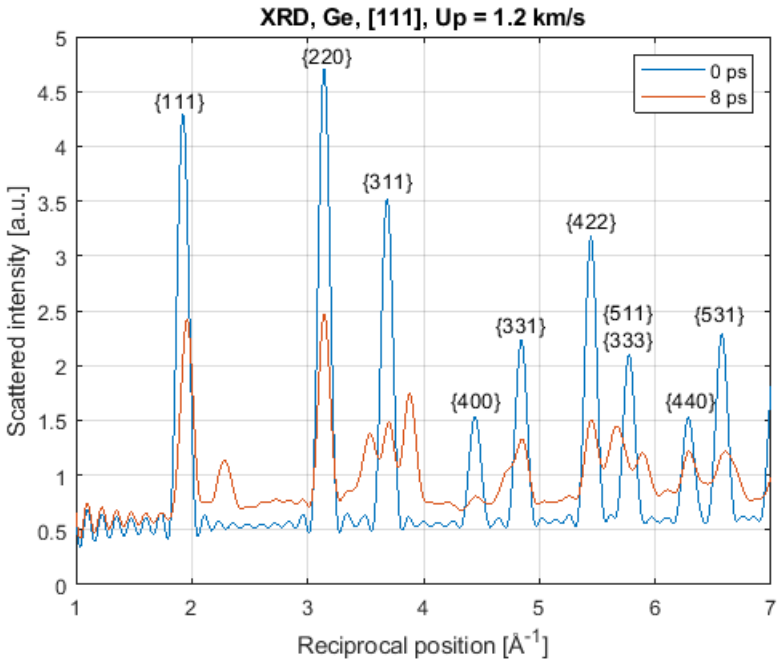

Figure 12: XRD pattern of germanium compressed in [111] with piston velocity 1.2 km/s. Crystalline germanium is in blue, shocked in red.

3.1.2. Compression along [111]

Germanium was also simulated compressed in [111] to compare to the compression in [001] and the compression in [111] present in the experiment in InSb. With a piston speed of 1.2 km/s along [111] in germanium, the entire shocked region is unable to be structurally identified despite stretching the crystal with affine transformations. The XRD for the unshocked and shocked material after 8 ps is shown in figure 12. The same trends as in the XRD pattern when compressing along [001] (figure 10) is seen, with the original crystalline peaks lowering, new emerging, and light scattering for all scattering vectors larger than 2 Å^{-1}. The strain shifted peaks emerge at roughly the same scattering vectors for both compression directions. The crystalline peaks before compression are broader in figure 12 than in figure 10. This has nothing to do with the compression direction but is a result of that the cut-off radius of the RDF was 80 Å in figure 10 and 50 Å in 12.

3.2. InSb

For all simulations in InSb the "Identify diamond structure" algorithm was unable to identify any structure of the shocked region, even when applying affine transformations beforehand. The material was stretched in the compression direction far beyond the actual compression of the simulation box of around 25 %. The algorithm was able to perfectly identify crystalline zinc-blende as well as wurtzite lattices, even when distorting the lattices by compressing 10 % and stretching 20 %.

3.2.1. Compression along [001]

To compare to the results in germanium along [001], InSb was first compressed in this direction. With a piston speed of 1.2 km/s, the velocity profile is shown in figure 13. Similarly to in germanium with a piston speed of 1.1 km/s (figure 4) the atoms past the shockfront travel with the piston speed. A dent of around 0.1 km/s occurs in the shockfront. This dent does not resemble the broader velocity reduction seen in germanium above the amorphization threshold in figures 9 and 11. Whether the dent corresponds to an actual feature or is a modeling error is unknown. The shockspeed is slower than the 5.42 km/s found in germanium. Estimating the shockspeed from where the velocity has reached 0.2 km/s gives an average shockspeed of 3.37 km/s.

The XRD of the unshocked material and the shocked material 8.75 ps later is shown in figure 14. No new distinct peaks emerge, and X-rays are scattered for all positions above 2.0 Å^{-1}. This is similar to the amorphous contribution seen for germanium with identical compression conditions in figure 10. A difference is that the strain shifted peaks seen in germanium do not appear in InSb. The lack of strain shifted peaks in InSb is consistent with that the "Identify diamond structure" algorithm does not find any zinc-blende structure even when un-straining the lattice along [001], as opposed to in germanium where cubic diamond is identified between the amorphous bands.

3.2.2. Compression along [111] with periodic boundaries

InSb was then compressed in [111] to relate to the experimental results. Building a zinc-blende or diamond cubic lattice in the cuboid simulation box oriented along $[10\bar{1}]$, $[\bar{1}2\bar{1}]$, and [111], no spacing such that the crystal planes are perfectly matched at the boundary in $(\bar{1}2\bar{1})$ were found. This problem was also present in the germanium simulation compressing in [111]. Figure

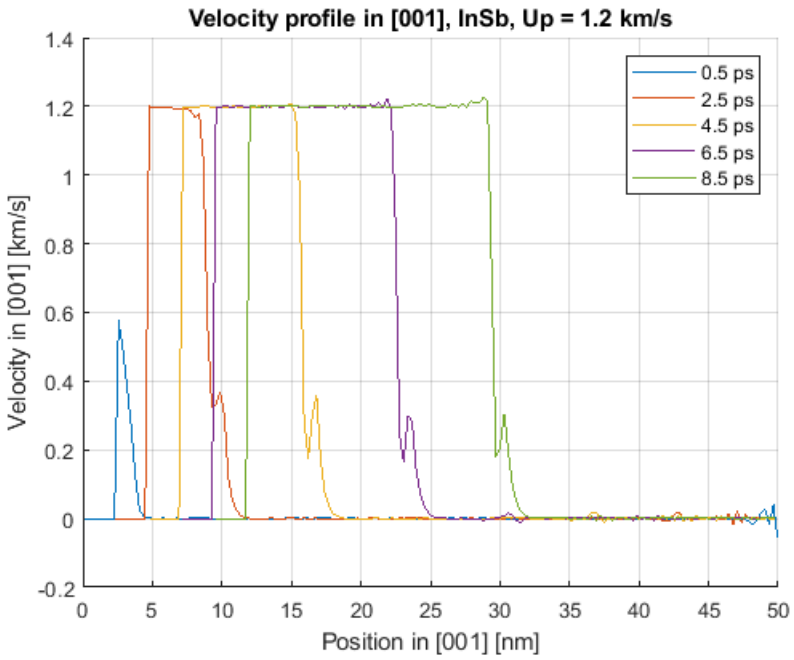

Figure 13: Velocity in InSb along [001] as a function of position for five different times. Piston speed is 1.2 km/s.

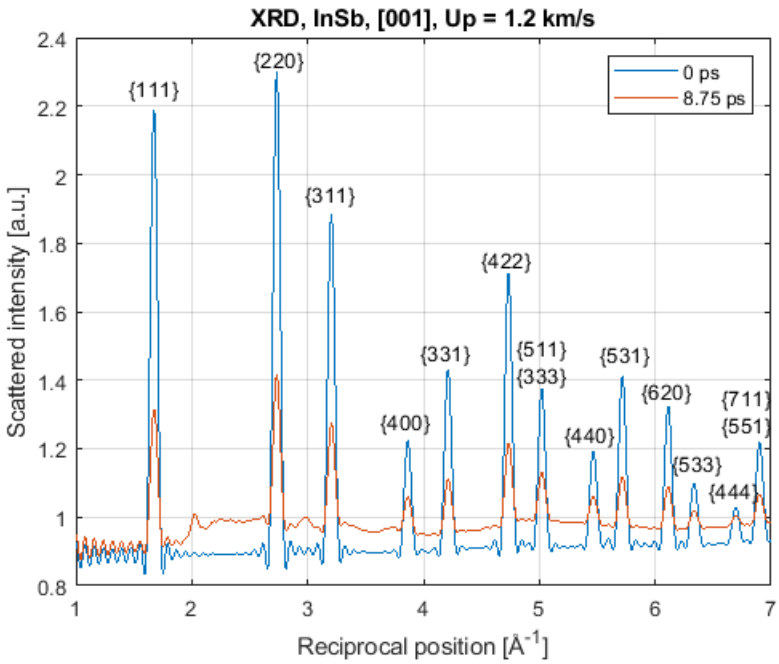

Figure 14: XRD pattern of InSb compressed in [001] with piston velocity 1.2 km/s. Crystalline InSb is in blue, shocked in red.

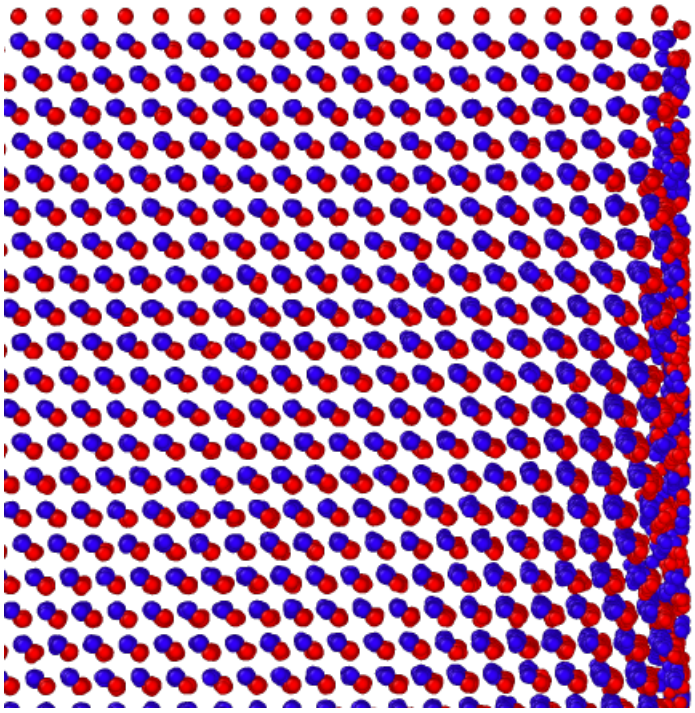

Figure 15: Image of the $[\bar{1}2\bar{1}][111]$ plane after equilibration with [111] upwards in figure. Indium atoms are in blue, antimony atoms in red.

15 illustrates the boundary surface in $(\bar{1}2\bar{1})$ after equilibration. The atoms at the boundary have larger displacements than the atoms in the bulk of the material, and there is tension in the surface. When the periodic boundary conditions along [111] is lifted after equilibration, the tension in $(\bar{1}2\bar{1})$ is partly relaxed along [111]. This is illustrated in figure 16 that shows the periodic $(\bar{1}2\bar{1})$ boundary to the right and the non periodic (111) boundary upwards, 1 ps after the equilibration. The excess energy in the $(\bar{1}2\bar{1})$ boundary is transferred to atoms in the (111) boundary. Some atoms with sufficiently high velocities escape the crystal completely.

The imperfect periodic boundary is a pure modeling error that does not relate to anything in the actual experiment. To avoid sampling the boundaries in the XRD, only the RDF of atoms in the central 20x20 nm was considered.

Compressing with a piston speed of 1.2 km/s, the atom velocities along

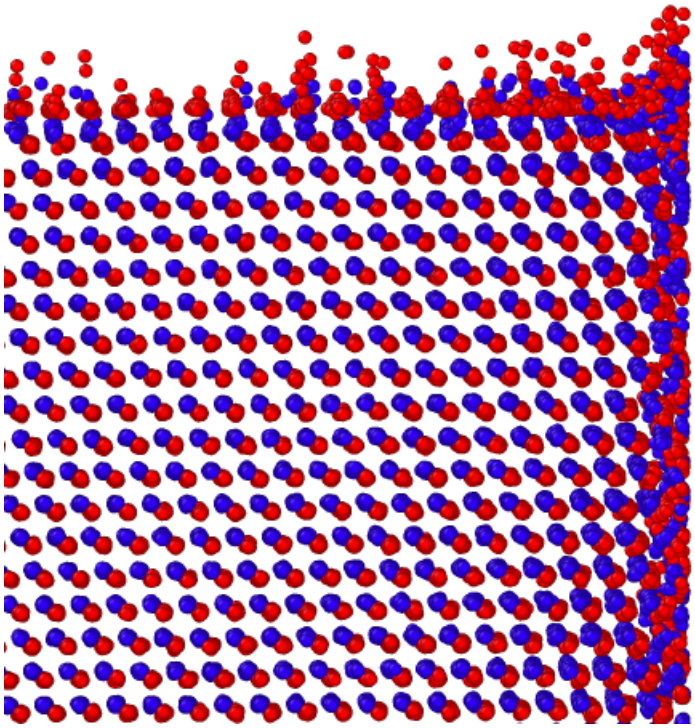

Figure 16: Image of the $[\bar{1}2\bar{1}][111]$ plane after 1 ps with [111] upwards in figure. Indium atoms are in blue, antimony atoms in red. Atoms escape the system in [111].

[111] is shown in figure 17. The figure is truncated at 60 nm to exclude the atoms escaping the crystal. Atoms past the shockfront accelerate to the piston velocity as when compressing in [001] (figure 13). The atoms adjacent to the piston oscillate around the piston velocity. At the shockfront a pre-pulse that also includes negative velocities can be seen. The oscillatory behaviour near the piston is likely a result of how the piston is modeled. A zincblende lattice can be viewed as a layered material along [111], with each layer consisting of two slightly separated layers of In and Sb. As the atoms in the piston is completely static in relation to each other, it is essential that the piston consists of entire layers of the lattice. Otherwise half a layer will sink into the piston and, with the attractive force of the half-layer that is part of the piston, be launched similarly to a slingshot. The problem can be resolved by selecting entire layers of the crystal to be part of the piston, which is implemented in the script in appendix E.3. Possibly the oscillation of this half-layer could be the cause of the pre-pulse seen in front of the shockwave.

The shock speed, estimated from where the particle velocities have reached 0.2 km/s, is on average 3.30 km/s. This is close to the simulated shock speed of 3.37 km/s for compression in [001]. However, it is well below the longitudinal speed of sound in [111] of 3.9 km/s [30]. This sound speed holds for zero pressure and a temperature of 300 K. It is therefore not clear that actual shockwaves have been simulated in InSb (though for consistency they will continue to be referred as such). The pressure and temperature dependence of the sound speed must be examined to determine whether or not it is a modeling error that the shock speed is only 3.30 km/s.

The XRD pattern is shown in figure 18. The original peaks are lowered while three new peaks emerge at 2.35, 3.61, and 4.53 Å^{-1}. The peak at 2.35 Å^{-1} is positioned far away from the adjacent {111} and {220} peaks, and therefore clearly is not a result of strain shifts. Above 5 Å^{-1} some X-rays are scattered for all scattering vectors. The relative positions of the shock induced peaks do not agree with where the strain shifted peaks appeared in germanium, figure 12. Furthermore, in germanium some X-rays are scattered for all positions above 2 Å^{-1} instead of at 5 Å^{-1}.

The 2.35 Å^{-1} peak in InSb is examined in more detail in figure 19 where the time evolution is shown in steps of 2 ps. The peak becomes stronger over time, due to the shockwave progressing through more material. In the observed timespan the peak does not shift in position.

Calculating the XRD from the RDF of the entire central 20x20 nm (as in

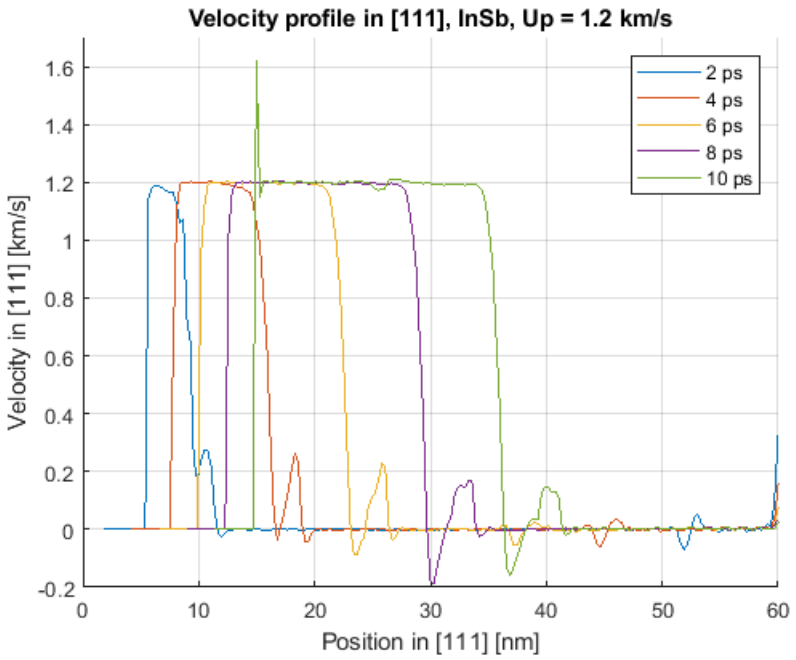

Figure 17: Velocity in InSb along [111] as a function of position (truncated at 60 nm) for five different times. Piston speed is 1.2 km/s.

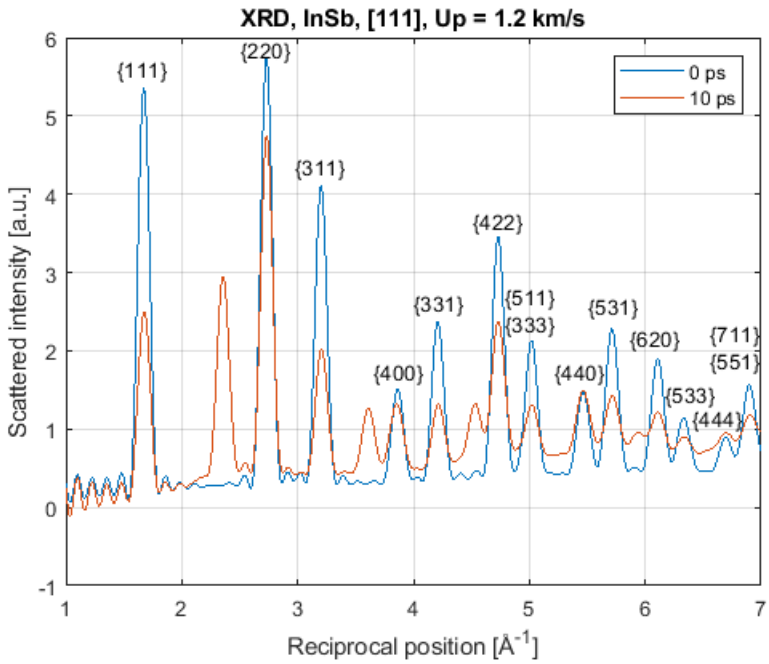

Figure 18: XRD pattern of InSb compressed in [111] with piston velocity 1.2 km/s after 0 and 10 ps. A distinct peak emerges at 2.35 Å^{-1}.

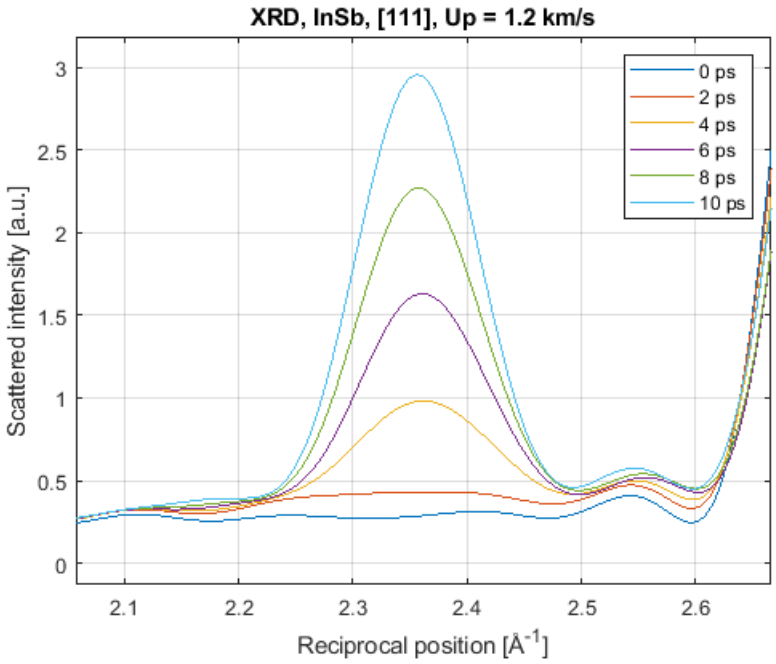

Figure 19: Zoom in on the 2.35 Å^{-1} peak in figure 18.

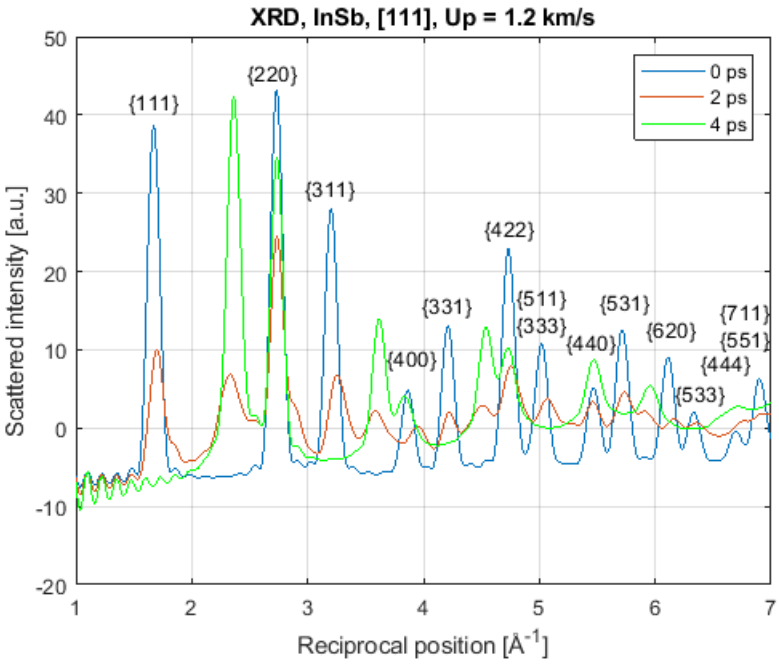

Figure 20: XRD pattern of InSb compressed in [111] with piston velocity 1.2 km/s taken for the chunk of material beneath the shockwave after 4 ps.

figure 18), both the unshocked material above the shockfront as well as the shocked material below is studied. Examining only the 10 nm closest to the piston along [111] in steps of 2 ps, the resulting XRD is shown in figure 20. The shockwave has partly propagated through the examined volume after 2 ps, and completely passed it after 4 ps. After the shockwave has passed the XRD pattern stays almost fixed. At 2 ps the transition between the peaks of the unshocked and shocked material is seen. The 2.35 Å^{-1} peak is the strongest feature in the completely shocked material.

The simulation was repeated several times with varying piston speeds to see how the XRD was affected and find under which conditions the 2.3 Å^{-1} peak appears. The XRD patterns for piston velocities of 1.0, 1.1, 1.2 and 1.3 km/s is shown after a duration of 8 ps in figure 21. (Note that the runs with piston speeds 1.1 and 1.2 km/s utilized a timestep of 1 fs, while the others used 2 fs). The original crystalline peaks appear at the same positions, with slightly varying intensity. The shock induced peak near 2.35 Å^{-1} varies both in intensity and position. Figure 22 shows this peak enlarged, from

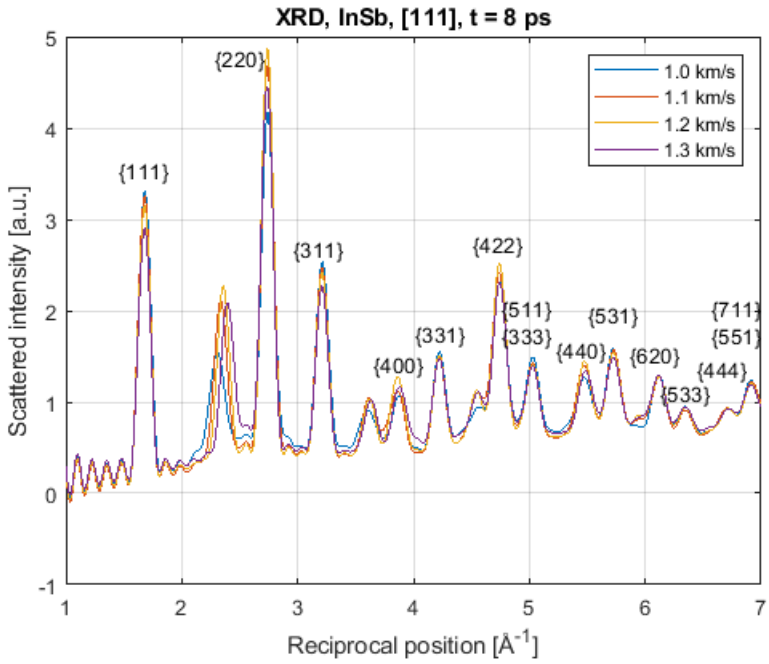

Figure 21: XRD pattern of InSb compressed in [111] at time 8 ps for piston velocities of 1.0, 1.1, 1.2, and 1.3 km/s.

which it is evident that the peak shifts towards higher scattering vectors for higher piston velocities.

A piston velocity of 1.0 km/s is the slowest simulated piston speed where a clear XRD peak emerges around 2.3 Å^{-1}. Figure 23 shows the time evolution of the XRD around 2.3 Å^{-1} in steps of 2 ps for a piston velocity of 0.95 km/s. The peak is broader and less intense compared to the larger piston velocities in figure 22, and the peak value shifts to lower scattering vectors over time.

3.2.3. Compression along [111] with non-periodic boundaries

As perfect periodic boundary conditions do not apply for the runs compressed in [111], an alternative system with non-peridic boundaries was also studied. Compressing the 40x40x60 nm crystal with non-periodic boundary conditions in all directions, the material partly relaxes by expanding normal to the compression direction. This is shown in figure 24, that shows a cross section of the material with the atoms coloured by von Mises stress between 0 and 12 GPa. The violent expansion normal to the compression causes some atoms to leave the crystal. The strain and stress is not completely uniaxial.

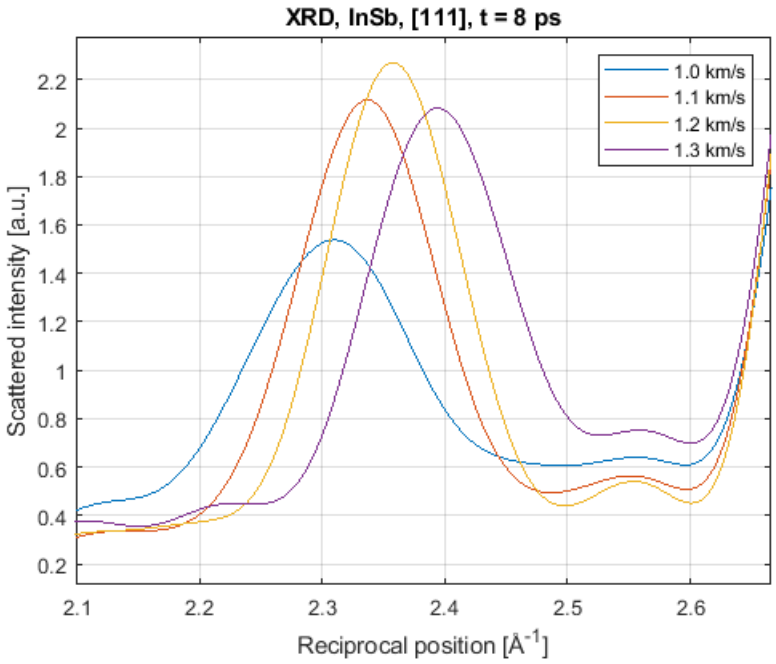

Figure 22: Zoom in on the 2.35 Å^{-1} peak in figure 21.

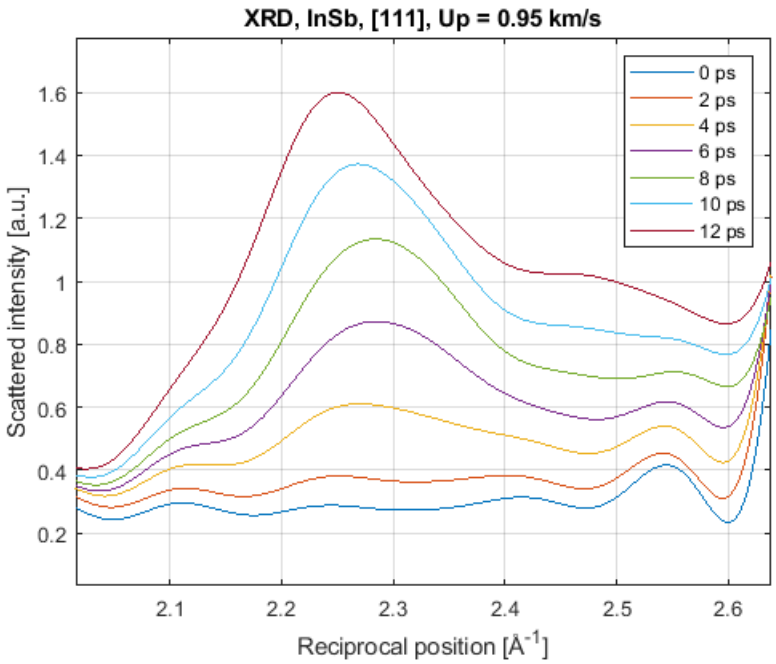

Figure 23: XRD pattern of InSb compressed in [111] with piston velocity 0.95 km/s for seven times in steps of 2 ps.

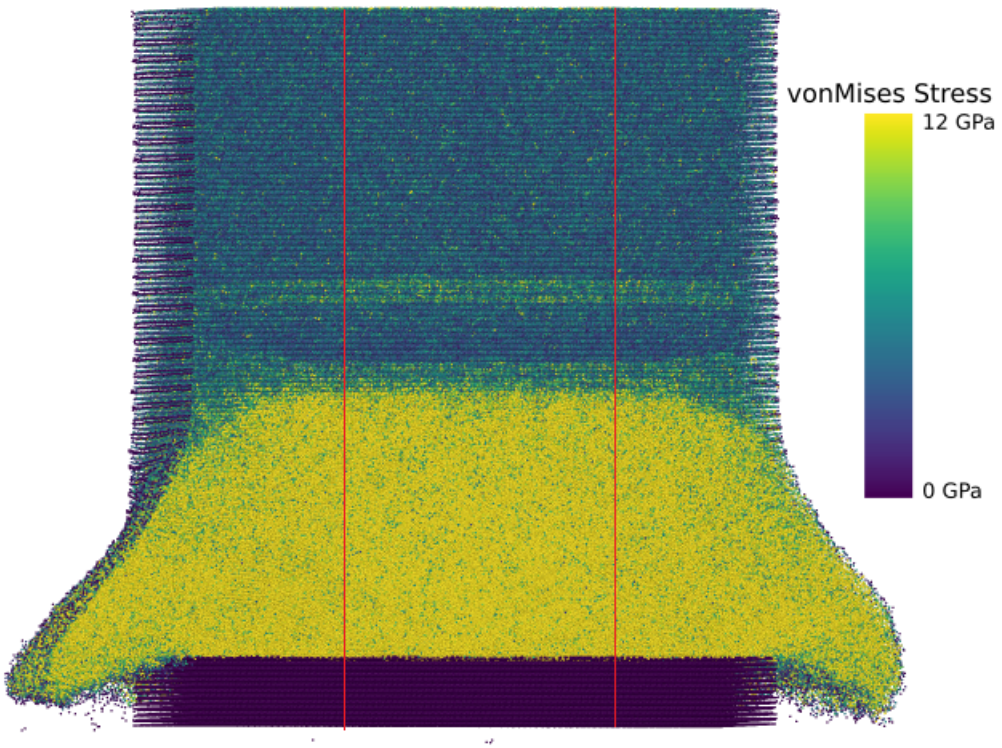

Figure 24: Cross-section of InSb compressed in [111] with non-periodic boundaries after 10 ps. Piston velocity is 1.2 km/s and atoms are coloured by von Mises stress between 0 and 12 GPa. The central 20 nm are indicated by red lines.

The shockfront has a slight curvature that reaches a maximum at the center of the crystal. The central 20x20 nm of the same simulation is shown in figure 25. In this block the shockfront is approximately plane. The relaxation normal to the compression direction causes the total number of particles in this region to decrease throughout the simulations.

The time evolution of the XRD for a piston speed of 1.2 km/s with non-periodic boundary conditions is shown in figure 26 after 0, 6, and 10 ps. The same peaks as in the XRD with periodic boundary conditions (figure 18) can be seen after 6 ps, including the three shock induced peaks near 2.3, 3.6, and 4.5 $Å^{-1}$. However they do not rise uniformly over time at the same scattering vector. The 2.3 $Å^{-1}$ peak lowers in intensity and the shape distorts after 10 ps, probably due to the relaxation of the material.

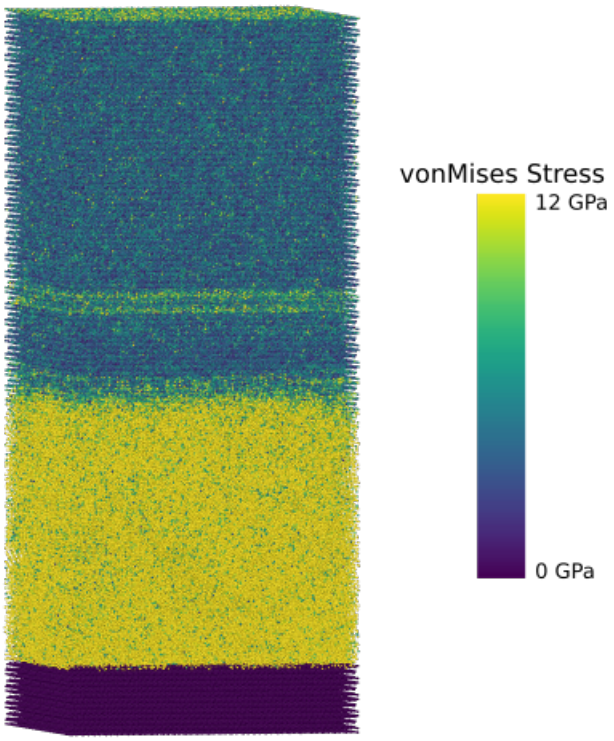

Figure 25: Illustration of only the central 20x20 nm in figure 24. Atoms coloured by von Mises stress between 0 and 12 GPa.

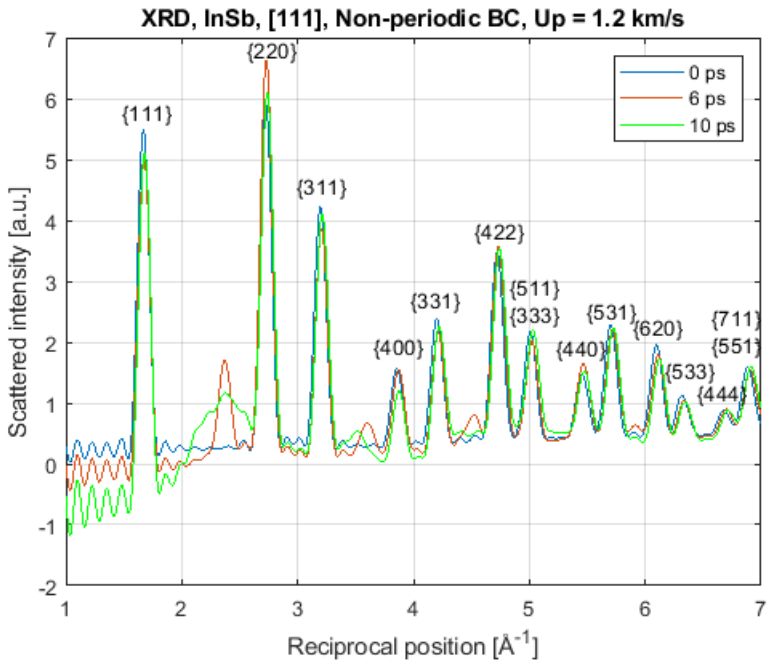

Figure 26: XRD pattern of InSb compressed in [111] with non periodic boundaries and piston velocity 1.2 km/s after 0, 6, and 10 ps.

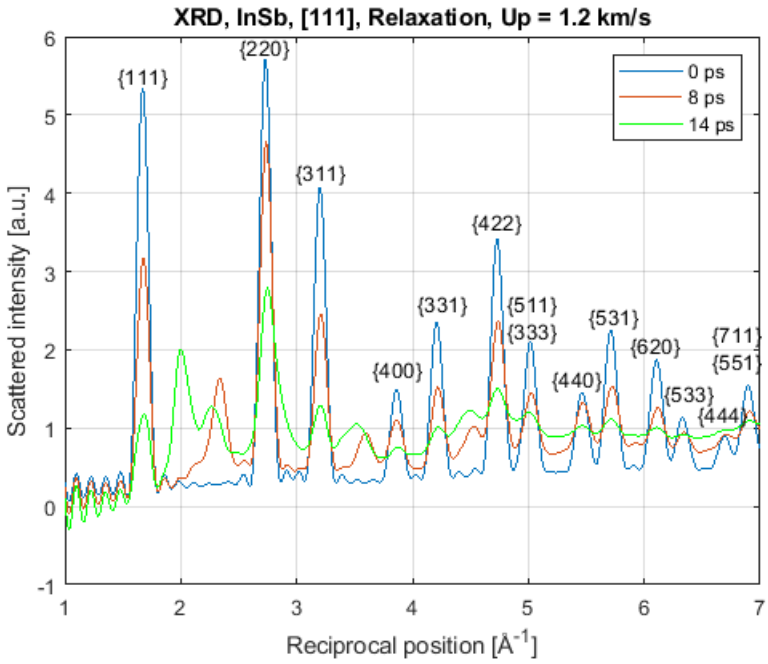

Figure 27: XRD pattern of InSb compressed and relaxed in [111] with piston velocity 1.2 km/s after 0, 8, and 14 ps.

3.2.4. Compression and relaxation along [111]

So far InSb has only been studied during compression. To examine what features remain after the material has relaxed, simulations with a decelerating piston was conducted on a longer timescale. Compressing with 1 ps piston acceleration, 5 ps fix velocity of 1.2 km/s, 2 ps deceleration, and 6 ps kept fixed, the resulting XRD pattern is shown in figure 27. As the material relaxes, the 2.3 Å^{-1} peak lowers and a new peak near 2.0 Å^{-1} emerges. Figure 28 shows this in more detail. This is in disagreement with the actual experiment where the 2.3 Å^{-1} peak remained in the relaxed material.

A modeling error occurred due to the deceleration over 2 ps being too rapid for the piston to halt the bulk material. The piston therefore became dislodged from the rest of the crystal. This behaviour corresponds poorly to the actual experiment in InSb, where the crystal of course stays intact. The dislodged piston enabled the material to relax both in [111] and [11$\bar{1}$]. The ability to relax in [11$\bar{1}$] should not have existed, and could possibly have affected the XRD.

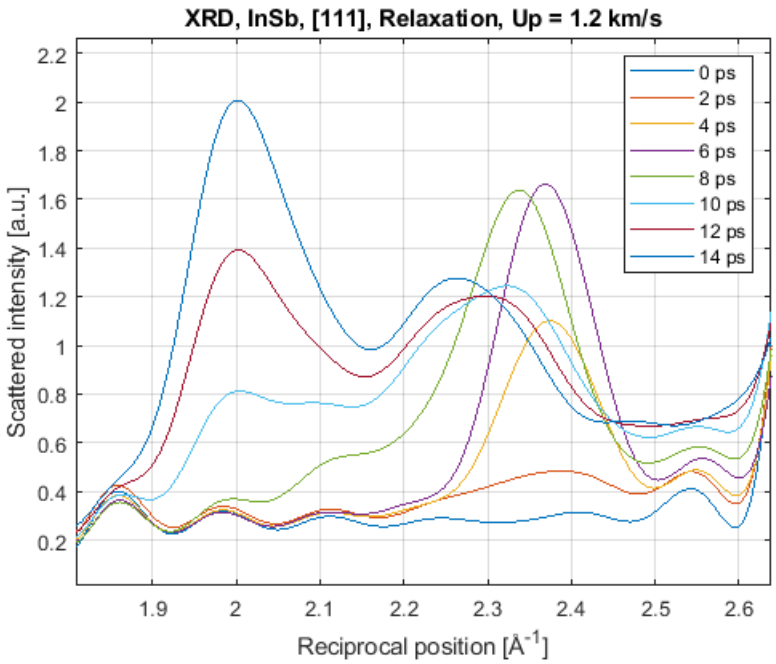

Figure 28: Zoom in of the 2.35 Å^{-1} in figure 27 in steps of 2 ps. The peak shifts to 2.0 Å^{-1} over time.

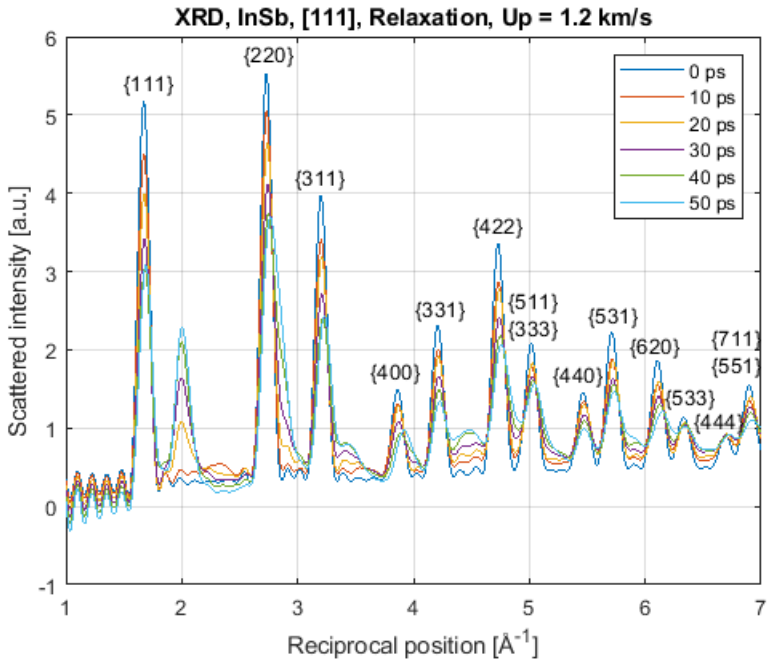

Figure 29: XRD pattern of InSb compressed and relaxed in [111] with piston velocity 1.2 km/s for six times in steps of 10 ps. A distinct peak at 2.0 Å^{-1} emerges.

Increasing the total simulation time from 14 ps to 50 ps with identical piston behaviour, the XRD pattern is shown in steps of 10 ps in figure 29. Scattered intensity around 2.3 Å^{-1} can only be seen at 10 ps, afterwards the peak is completely missing. Meanwhile the peak at 2.0 Å^{-1} grows stronger throughout the entire simulation.

3.3. Runs with no compression

As the experimental XRD was taken through the aluminium layer, it is relevant to compare the XRD to that of aluminium. The XRD pattern of crystalline aluminium at a temperature of 300 K is shown in figure 30 alongside the peaks of InSb that has been compressed for 12 ps with a piston speed of 1.2 km/s with imperfect periodic boundary conditions. The aluminium peaks do not coincide with the shock induced peaks near 2.3, 3.6, and 4.5 Å^{-1}.

To examine the possible hypothesis that the 2.3 Å^{-1} peak stems from a wurtzite structure, the XRD pattern of a wurtzite InSb crystal at a temperature of 300 K was studied. It is shown in figure 31. While there is a clear

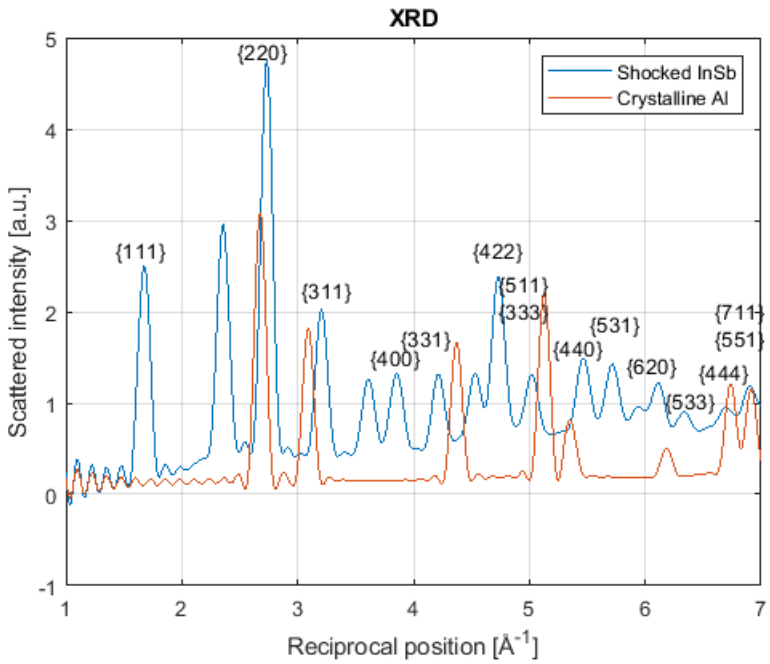

Figure 30: XRD pattern of an aluminium crystal at temperature 300 K compared to the pattern in InSb compressed in [111] after 10 ps with piston velocity 1.2 km/s.

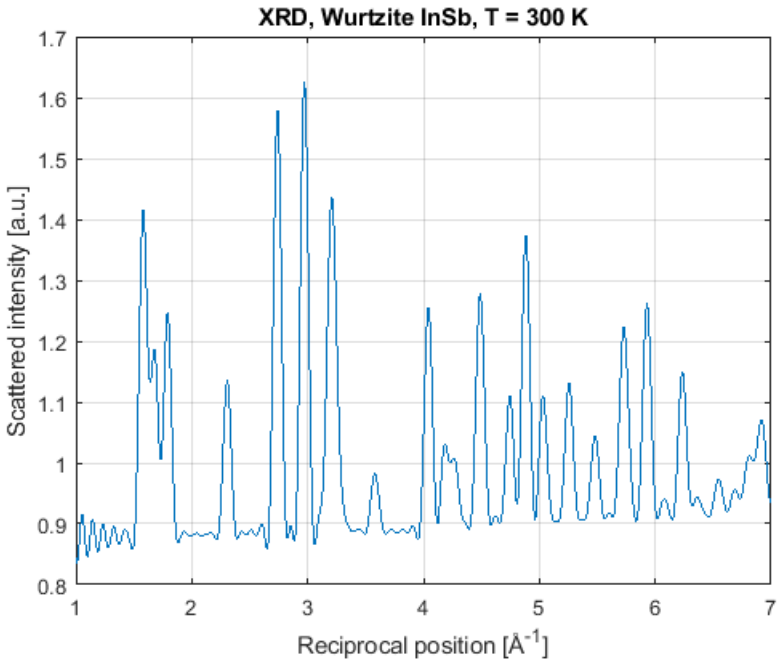

Figure 31: XRD pattern of a wurtzite InSb crystal at 300 K.

peak at 2.3 Å^{-1}, there are several peaks not seen in the crystalline or shocked zincblende InSb. Examples of such peaks are at 2.96, 4.05 and 4.88 Å^{-1}.

3.4. Runs examining model validity

Part of the effect of the imperfect boundaries on the XRD was examined by calculating the XRD from RDF:s at different locations in the crystal. Figure 32 shows the XRD for, InSb compressed in [111], with a piston velocity of 1.1 km/s after 12 ps. In the blue curve the 20x20 nm section examined is in the center of the 40x40 nm simulation domain, and in the red curve the 20x20 nm section examined has its faces aligned with the $(10\bar{1})$ and $(\bar{1}2\bar{1})$ boundaries. The two patterns display the same number of peaks at the same positions, and overall the intensities are very similar. The largest difference in intensity is seen between the 2.3 Å^{-1} and the {220} peak, where the intensity between the peaks are higher and the peaks lower for the XRD taken at the edges of the simulation domain.

To investigate if the simulations with timesteps of 1 fs and 2 fs are comparable, two simulations with otherwise identical conditions were conducted.

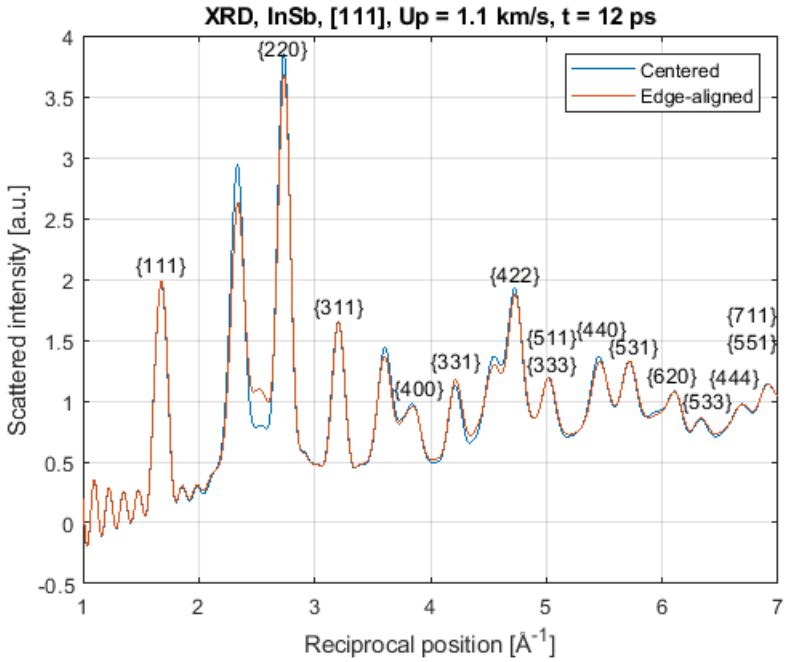

Figure 32: XRD patterns of InSb compressed in [111] with piston speed 1.1 km/s after 12 ps. The blue curve is extracted from the RDF of the central 20x20 nm, while the red curve is from the RDF of a 20x20 nm cuboid with its faces at the simulation box boundaries.

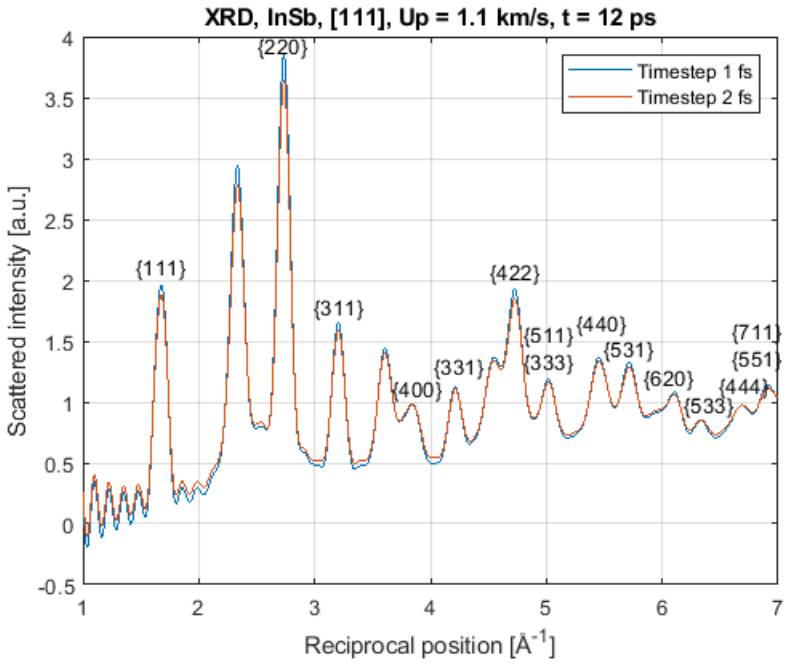

Figure 33: XRD pattern of InSb compressed in [111] with piston velocity 1.1 km/s simulated with timesteps of 1 fs (in blue) and 2 fs (in red).

The XRD after 12 ps compression with a piston velocity of 1.1 km/s is shown in figure 33. If the timesteps are comparable, the XRD should ideally be identical. A slight difference in scattered intensity can be seen for the peak values, although the overall characteristics of the two runs are very similar. The 2.35 Å^{-1} peak is stronger for the shorter timestep. Whether or not the observed intensity difference holds in general can not be said without knowing the variance of the intensity between identical runs.

4. Discussion

4.1. Relation to experimental measurements in InSb

During compression in InSb along [111] with piston velocities above 1.0 km/s, an XRD peak emerges between 2.3 and 2.4 Å^{-1} using both periodic and non-periodic boundary conditions. As shown in figure 20 this peak is the strongest feature of the shocked material. This is in good agreement with the actual experiment where a peak at 2.3 Å^{-1} was observed. That the XRD peaks of aluminium in figure 30 do not coincide with the peaks in shocked

InSb suggests that taking the experimental XRD measurement through the aluminium did not hinder resolving the features of InSb.

That the exact position of the 2.3 $Å^{-1}$ peak in figure 22 varies with piston velocity is likely due to strain shifting. (A higher piston velocity gives a larger strain and thus shorter distances between reflecting planes, which corresponds to larger scattering vectors). That the peak no longer forms below piston velocities of 1.0 km/s suggests that the 2.3 $Å^{-1}$ peak is not simply the [111] peak at 1.7 $Å^{-1}$ that has been strain shifted very far.

The structural origin of the 2.3 $Å^{-1}$ peak remains unclear. From equation (3) we know it corresponds to an atom spacing of 2.7 Å present in the material. While the XRD of the crystalline wurtzite InSb in figure 31 display an XRD peak at 2.30 $Å^{-1}$, the lack of other wurtzite peaks in the experimental measurement and shock simulations suggests no wurtzite InSb forms in the simulations. This is further implied by that no hexagonal diamond structure is found by the "Identify diamond structure" algorithm even when stretching the lattice. The algorithm is able to perfectly identify such a structure for every atom in the lattice built as a wurtzite structure.

Parallels can be drawn to lonsdaleite (hexagonal diamond), an allotrope of carbon that is thought to form from graphite under high pressure conditions. Kraus et al. report the formation of lonsdaleite based on diffraction measurements alone [31]. This is questioned by Németh et al., that with single transmission electron microscopy (STEM) images in meteorites, thought to contain lonsdaleite, find no evidence of the material. Instead they find multiple stacking faults and twinning that they with structure models show display the characteristic spacings that give rise to the XRD peaks attributed to lonsdaleite [32].

Whether any of the stacking faults and twinnings found by Németh et al. can be found in the simulated InSb is the next step in explaining the origin of the 2.3 $Å^{-1}$ peak. Assuming that the distances in the faults scale proportionally with the lattice constant of the original crystal, the 1.16 Å spacing described by Németh et al. [32] corresponds best to the 2.7 Å spacing in InSb that gives the 2.3 $Å^{-1}$ peak. The spacing in carbon arises due to a stacking fault in {111} visible in STEM images in [011] [32].

In the simulations where the shocked material is able to relax, the 2.3 $Å^{-1}$ peak disappears and a new peak at 2.0 $Å^{-1}$ emerges instead. Whether this is a modeling error caused by the dislodged piston should be examined with simulations where the piston decelerates slower than over 2 ps. Ideally the piston should closely mimic the behaviour of the expanding aluminum. A

more refined model could possibly be achieved by abandoning the piston methodology and instead incorporating a portion of the aluminium. However, including the laser heating would require additional modeling considerations outside the MD framework, as a classical MD simulation does not model the heat transfer between the atoms and the free electron gas (the electron-phonon coupling) [11].

Most simulations had a total duration of 12 ps, with the longest extending to 50 ps. This is much shorter than the observed timespan in the experimental XRD-measurement (up to several μs). Simulating longer timescales (even over a single nanosecond) is not computationally feasible with a system of this size. One approach to model relaxation over slightly longer timescales (0.5 ns) would be to use a longer timestep after the shockwave has dissipated into the crystal.

4.2. Model validity

The minimal difference on the XRD in figure 33 between using a timestep of 1 fs and 2 fs respectively shows that the resulting peak positions between simulations with different timesteps are comparable. To tell whether the slight variations in intensity stem from the different timesteps or not, the variance in intensity between successive runs with identical simulation conditions has to be examined. The lowest timestep is to be preferred for greater accuracy. However, to decrease the timestep further would require more computer resources or decreasing the system size.

The XRD peaks for the simulations, where the cut-off radius of the RDF was 80 Å (such as in figure 14), are sharper than the peaks from simulations with a cut-off radius of 50 Å (such as in figure 18). This is due to that a larger cut-off radius makes the truncation of the sine-transform in equation 5 less severe; including more frequencies in the transform enables sharper features to be resolved. This suggests that the width of the XRD peaks in all simulations with a cut-off radius of 50 Å is limited by this truncation, and not temperature broadening.

The computational effort in calculating the RDF for large radii is substantial. In the germanium system with four million atoms, cut-off radii of 50, 70, and 80 Å takes 24, 51, and 110 minutes to calculate. If XRD simulations with larger cut-off radii is desired, it could be worthwhile to look into parallelizing this calculation as well. The system size also sets an upper limit on the possible radii.

The XRD measurement in figure 32 taken from different parts of InSb with imperfect periodic boundary conditions exhibits the same principal features in terms of the number of peaks and their position. From this it is plausible that the chaotic behaviour near the boundaries seen in figure 15 do not directly influence the XRD considerably, verifying that the large atom displacements are a local feature to the boundaries.

While the imperfect periodic boundaries are not part of the central 20x20 nm used for the XRD measurements, they still influence all atoms in the crystal. Due to the high displacements and velocities of the atoms at the boundary, they also have a larger temperature than the interior of the crystal. As the entire system is equilibrated to a temperature of 300 K, the interior atoms that are examined is actually at a lower temperature.

The effect of the imperfect boundaries on the crystal and possible defect formation is difficult to estimate. For truly reliable results when compressing in [111], a model with perfect periodic boundary conditions must be constructed. This may or may not be possible in a zincblende lattice with a rectangular cuboid simulation box. However, with a triclinic simulation box it should be possible. This would require reworking parts of the script as there are some commands in LAMMPS that only function with the default simulation box shape. For example, orienting a triclinic simulation box in [100], [010], and [111] a periodic system could be created. A better chosen unit cell than the cubic unit cell used could be another potential way to construct perfect periodic boundaries.

The simulations in InSb with non-periodic boundaries do not display unphysical behaviour at the boundaries. However the non-uniaxial strain makes the system harder to analyze, as displacements, velocities and stresses in all directions must be considered. Furthermore the systems translates poorly to the conditions in the actual experiment where a shockwave is sent into a bulk material, not a 40x40 nm slab of InSb. Relaxation normal to the compression direction should not be possible. Increasing the simulation size would make the correspondence to the actual experiment better, but would require substantial computational effort.

A minor problem present in all simulations arises when the periodic boundary conditions in the compression direction is lifted after the equilibration. This causes a slight expansion of the top of the crystal that sends a sound-wave downwards through the crystal. This can for instance be seen as a small pulse near 60 nm after 0.5 ps in figure 4. It likely has little effect on the simulation as the amplitude is negligible compared to the shockwave.

The modified Noose-Hoover thermostat with dampened temperature and pressure oscillations is problematic, as a Maxwell-Boltzmann velocity distribution might not be achieved after equilibration even if this is the initial velocity distribution. For the production runs, the damping term in the thermostat should have been completely removed to create a crystal with the correct velocity distribution. This would require the equilibration running for a longer duration. The additional computational cost could be partly mitigated by using a longer timestep for the equilibration.

To further optimize the script in terms of computation time, the equilibration and shock sequence could be separated into two scripts. The first script would create the lattice, run an equilibration, and dumps the atom positions and velocities. The second script could then read this file, and immediately run the shock sequence. Such an implementation would only require running one equilibration for each lattice type and orientation, but would loose the ability to change the randomly assigned initial velocities between runs.

5. Conclusions

The developed LAMMPS script produces shock amorphization in germanium during compression along [001] consistent with the results published by Zhao et al. [3]. Simulated XRD patterns in germanium show little resemblance to the XRD patterns in InSb for both compression along [001] and [111]. Simulations with both imperfect periodic and non-periodic boundary conditions confirm the experimental result that an XRD peak around 2.3 Å^{-1} appears when compressing InSb along [111]. Furthermore the results confirm that crystalline aluminium should not have prevented resolving the XRD peaks of the shocked InSb. The structural origin of the 2.3 Å^{-1} peak remains unclear. Structure identification algorithms as well as XRD simulations of crystalline wurtzite InSb show no evidence of the peak stemming from the formation of wurtzite InSb. In simulations on a longer timescale with relaxation of the material, the 2.3 Å^{-1} peak disappears and a new peak at 2.0 Å^{-1} emerges. This is inconsistent with the experimental results where the 2.3 Å^{-1} peak remained after relaxation.

Further work is needed on post-processing of the results in OVITO to find the structure causing the XRD peak. Additionally the boundary conditions of the model must be examined in more detail, to confirm whether or not periodic boundary conditions when compressing along [111] is truly

impossible for all crystal orientations, or to rework the simulation to utilize a triclinic simulation box with perfect periodic boundaries.

Bibliography

[1] L.Q. Huston, A. Lugstein, Shen Guoyin, D.A. Cullen, B. Haberl, J.S. Williams, and J.E. Bradby. Synthesis of novel phases in si nanowires using diamond anvil cells at high pressures and temperatures. *Nano Letters*, 21(3):1427 – 1433, 2021.

[2] B.L. Holian, W.G. Hoover, B. Moran, and G.K. Straub. Shock-wave structure via nonequilibrium molecular dynamics and navier-stokes continuum mechanics. *Physical Review A (General Physics)*, 22(6):2798 – 2808, 1980.

[3] Shiteng Zhao, Bimal Kad, E. Wehrenberg Christopher, A. Remington Bruce, N. Hahn Eric, L. More Karren, and A. Meyers Marc. Generating gradient germanium nanostructures by shock-induced amorphization and crystallization. *Proceedings of the National Academy of Sciences of the United States of America*, 114(37):9791 – 9796, 2017.

[4] Steve Plimpton. Fast parallel algorithms for short range molecular dynamics. *Journal of Computational Physics*, 117:1–9, 1995. URL `http://lammps.sandia.gov`.

[5] Alexander Stukowski. Visualization and analysis of atomistic simulation data with OVITO-the Open Visualization Tool. *Modelling and simulation in materials science and engineering*, 18(1), JAN 2010.

[6] S. M. Sze and M. K. Lee. *Semiconductor Devices Physics and Technology, 3rd edition*, pages 1,3,19–23. Wiley, 2012.

[7] Deepak Anandan, Venkatesan Nagarajan, Ramesh Kumar Kakkerla, Hung Wei Yu, Hua Lun Ko, Sankalp Kumar Singh, Ching Ting Lee, and Edward Yi Chang. Crystal phase control in self-catalyzed insb nanowires using basic growth parameter v/iii ratio. *Journal of Crystal Growth*, 522:30 – 36, 2019.

[8] Robert E. Dinnebier and S. J. L. Billinge. *Powder Diffraction: Theory and Practice.*, pages 464,468–471. Royal Society of Chemistry, 2008.

[9] B. E. Warren. *X-ray diffraction.* Addison-Wesley publishing company, 1969.

[10] David A. Keen. A comparison of various commonly used correlation functions for describing total scattering. *Journal of Applied Crystallography*, 34:172–177, 2001.

[11] Gilead B. Tadmor and Ronald E. Miller. *Modeling materials : continuum, atomistic, and multiscale techniques.*, pages 321–328, 492–499, 504–505. Cambridge University Press, 2011.

[12] Ben Leimkuhler and Charles Matthews. *Molecular Dynamics With Deterministic and Stochastic Numerical Methods*, pages 8–18. Interdisciplinary Applied Mathematics: 39. Springer International Publishing, 2015.

[13] Sandia. Lammps documentation, pair_style eam command, Retrieved 22-06-2021. URL `https://docs.lammps.org/pair_eam.html`.

[14] Sandia. Lammps documentation, pair_style tersoff command, Retrieved 22-06-2021. URL `https://docs.lammps.org/pair_tersoff.html`.

[15] Sandia. Lammps documentation, pair_style vashishta command, Retrieved 22-06-2021. URL `https://docs.lammps.org/pair_vashishta.html`.

[16] Sandia. Lammps documentation, fix nve command, Retrieved 06-06-2021. URL `https://docs.lammps.org/fix_nve.html`.

[17] Sandia. Lammps documentation, fix nvt command, Retrieved 06-06-2021. URL `https://docs.lammps.org/fix_nh.html`.

[18] Sandia. Lammps documentation, compute stress/atom command, Retrieved 25-05-2021. URL `https://lammps.sandia.gov/doc/compute_stress_atom.html`.

[19] Albrecht Bertram and Rainer Glüge. *Solid Mechanics. Theory, Modeling, and Problems.* Springer International Publishing, 2015.

[20] D. Fabregat-Traver, A.E. Ismail, and P. Bientinesi. Accelerating molecular dynamics codes by performance and accuracy modeling. *Journal of Computational Science*, 27:79, 2018.

[21] M. Chollet, R. Alonso-Mori, M. Cammarata, D. Damiani, J. Defever, J.T. Delor, Feng Yiping, J.M. Glownia, J.B. Langton, S. Nelson, K. Ramsey, A. Robert, M. Sikorski, Song Sanghoon, D. Stefanescu, V. Srinivasan, Zhu Diling, H.T. Lemke, and D.M. Fritz. The x-ray pump-probe instrument at the linac coherent light source. *Journal of Synchrotron Radiation*, 22(3):503, 2015.

[22] Oscar Guerrero-Miramontes. A beginner's guide to the modeling of shock/uniaxial/quasi-isentropic compression using the lammps molecular dynamics simulator, 2013. URL `https://www.researchgate.net/publication/259644293\_A\_beginner's\_guide\_to\_the\_modeling\_of\_shockuniaxialquasi-isentropic\_compression\_using\_the\_LAMMPS\_molecular\_dynamics\_simulator\_By\_Oscar\_Guerrero-Miramontes`.

[23] Hahn E. N., Zhao S., Bringa E. M., and Meyers M. A. Supersonic dislocation bursts in silicon. *Scientific Reports*, 6:5–6, 2016.

[24] H.J.C. Berendsen, J.P.M. Postma, W.F. van Gunsteren, A. DiNola, and J.R. Haak. Molecular dynamics with coupling to an external bath. *Journal of Chemical Physics*, 81(8):3684 – 3690, 1984.

[25] E. Maras, O. Trushin, A. Stukowski, T. Ala-Nissila, and H. Jónsson. Global transition path search for dislocation formation in ge on si(001). *Computer Physics Communications*, 205:13 – 21, 2016.

[26] Rune Alphonce and Helen Pilström. *Formler och Tabeller från Natur Kultur*, page 21. Natur Kultur, 2011.

[27] J. Tersoff. Modeling solid-state chemistry: Interatomic potentials for multicomponent systems. *Physical Review B*, 39(8):5566–5568, 1989.

[28] José Pedro Rino, Giovano de Oliveira Cardozo, and Adalberto Picinin. Atomistic modeling of the structural and thermal conductivity of the insb. *Computers, Materials & Continua*, 12(2):145–156, 2009.

[29] Y. Mishin, D. Farkas, M.J. Mehl, and D.A. Papaconstantopoulos. Interatomic potentials for monoatomic metals from experimental data and ab initio calculations. *Physical Review B - Condensed Matter and Materials Physics*, 59(5):3393–3407, 1999.

[30] L.J. Slutsky and C.W. Garland. Elastic constants of indium antimonide from 4.2°k to 300°k. *Physical Review*, 113(1):167–169, 1959.

[31] Kraus D., Ravasio A., Gauthier M., Gericke D. O., Vorberger J., Frydrych S., Helfrich J., Fletcher L. B., Schaumann G., Nagler B., Barbrel B., Bachmann B., Gamboa E. J., Göde S., Granados E., Gregori G., Lee H. J., Neumayer P., Schumaker W., Döppner T., Falcone R. W., Glenzer S. H., and Roth M. Nanosecond formation of diamond and lonsdaleite by shock compression of graphite. *Nature Communications*, 7(1):1 – 6, 2016.

[32] P. Nemeth, P.R. Buseck, L.A.J. Garvie, T. Aoki, N. Dubrovinskaia, and L. Dubrovinsky. Lonsdaleite is faulted and twinned cubic diamond and does not exist as a discrete material. *Nature Communications*, 5, 2014.

Appendix A. Guide to LAMMPS

This section descries the basics of how to set up and run LAMMPS. The structure of an input script, as well as specific commands will be discussed on such a level that the input scripts used in this work is decipherable. More extensive information can be found in the LAMMPS-documentation, available through https://lammps.sandia.gov/doc/.

Appendix A.1. Running LAMMPS

First LAMMPS has to be downloaded and installed. Installation files and guides on how to set up are available through https://lammps.sandia.gov/. LAMMPS is run from the command line with instructions read from a separate file, the input script. The input script is a file created by the user in a regular text-editor, such as "Notepad" on Windows or "Nano" on Linux. Convention is to save an input script with the extension ".in", e.g. "myFilename.in". Note that some text-editors will automatically put a ".txt" extension if it is not explicitly avoided by changing the file-format to "All files".

LAMMPS runs in the directory where the input script is located. This is also where the output data is written to by default. Navigate to this directory in the command line, and run LAMMPS by issuing a command on the form "executable -in myInputScript.in". The name of the LAMMPS executable varies slightly depending on what version is installed, and whether LAMMPS should be run in parallel or only in serial. To find the name of the executable,

search the "bin"-folder of the LAMMPS installation. With the "9Oct2020" version on Windows, "lmp_serial" works, and on Aurora simply "lmp" is the executable.

Some useful commands for navigating directories in the UNIX-command line are:

"cd path/to/file". cd stands for "Change Directory". Changes the current directory to the directory specified after "cd". All movement between directories is relative to the current location. Therefore the argument must be a sub-directory to the current directory. To avoid issuing multiple "cd" commands when moving to subsequent sub-directories, several arguments can be chained after one another with "/". To move up to the parent directory, use the argument "..".

"ls" (Linux) or "dir" (Windows). Stands for "list" and "directory" respectively. Prints all sub-directories and files accessible from the current directory.

"pwd" (Linux). Stands for "Print work directory". Prints the location of the current directory. On Windows the current directory is automatically displayed in front of the command prompt.

Most UNIX-commands have additional options. These are specified as a "-" with a keyword after. For instance, "ls -F" marks what elements are directories (folders) with a "/". This is the logic behind the command where LAMMPS is executed: The "-in" keyword tells the executable that it should read input from the file specified afterwards.

More advanced tasks such as copying, deleting, and sorting files from the command line, chaining together multiple commands in a UNIX-script, and feeding additional input parameters to the LAMMPS input script from the command line, are also possible.

Appendix A.2. LAMMPS input script structure

Every line in the input script contains a command name, followed by arguments specific to that command's syntax. The syntax and possible arguments for each command are listed in the documentation, see https://lammps.sandia.gov/doc/commands_list.html. LAMMPS reads the input script one line at a time, executing each command in turn. Comments are made with "#". LAMMPS jumps to the next line whenever this character is encountered, so comments can be made at the end of a command on the same line. When LAMMPS reaches the end of the script it automatically

exits. Some commands requires others to be previously set, meaning that every LAMMPS script typically follows the same outline:
- Initialize the simulation.
- Define variables.
- Create the simulation box.
- Create atoms and subdivide them into groups.
- Define the potential.
- Define computes and fixes for MD.
- Define computes and fixes for outputs.
- Run.
- Repeat defining fixes and running if needed.

Appendix A.3. Initialization and user-defined variables

Initialization sets up general properties of the simulation that need to be set before atoms and the simulation box is created. These are the amount of dimensions of the simulation, units used, initial boundary conditions, and what attributes to associate with the atoms. Relevant commands are "dimension", "units", "boundary", "atom_style" and "atom_modify". A typical choice of units for solid-state MD is "units metal", which uses distances in Å, time in ps, and energies in eV. The timestep is set with "timestep".

"boundary" determines the simulation box behaviour at the boundaries. It takes three arguments, for each dimension respectively. "p" gives periodic boundaries, meaning that atoms at the edges interact with atoms at the other side of the box, and atoms passing through the surface will wrap around to the other side of the simulation box. The other three options, "s", "f" and "m" all give non-periodic boundaries. For "s" the simulation box dimension is shrink wrapped to exactly encompass all atoms. "m" behaves similarly but with simulation box dimensions lower bounded by their initial size. "f" gives a fix simulation box boundary, where atoms that exit the boundaries are lost.

Defining the variables that are modified between runs right after the initialization is good practice. Variables are defined with the "variable" command. Numeric variables are defined on the form "variable name equal value", and strings on the form "variable name string "stringValue"". Entire mathematical expressions can be entered enclosed in "". To reference the value of a variable in later commands, the syntax "${variableName}" or

"v_variableName" is used. In the former, the variable is evaluated once immediately. In the latter, the variable is evaluated whenever its value needs to be used. Commands that write to a file or the screen several times requires the latter form.

Appendix A.4. Creating a simulation box and atoms

Creating atoms first requires the simulation box to be created. Regions in space are selected with the "region" command. The simulation box is created with "create_box", with a region as an argument. Points in space are selected with the "lattice" command, and atoms can be created at these sites with "create_atoms". Atoms can be subdivided into groups with the "group" command, and later referenced together.

When setting up the simulation box with "create_box", one must specify the number of atom types that is to be used in the simulation. This number can be greater than the number of elements used in the model, though atoms of the same type must be of the same element. In the Ge model, two atom types are used to clearly distinguish between the atoms in the piston and in the bulk material. A group of atoms can be set to type N by the statement "set group group-id type N".

With "lattice", basis vectors and the positions of the basis atoms in a unit cell are defined. Some crystal structures such as fcc, bcc and diamond are built into LAMMPS, while building other structures requires using the "custom" style. The "origin" keyword shifts the origin in fractions of a unit cell. The "orient" keyword maps the crystallographic axes to the simulation box axes, e.g. issuing "orient x 1 0 -1 orient y -1 2 -1 orient z 1 1 1" will select a lattice with the 111-crystallographic direction aligned with the z-axis. The selected points are automatically replicated infinitely in all directions, so to create atoms in the entire simulation box only one unit cell needs to be specified.

"lattice" can be used before the simulation box is created. This is advantageous when constructing a periodic model, as the simulation box dimensions is then an integer number of the lattice spacing. LAMMPS automatically calculates the extent of the unit cell along each direction regardless of its orientation. Appending the keywords "units lattice" to the "region" command selects a region based on this scale.

With a lattice selected, "create_atoms 1 box" creates atoms (of type 1) in the entire simulation box. The "box" keyword ensures LAMMPS places strictly one, not two or zero, atoms on periodic boundaries. Alternatively

"create_atoms 1 region region-id" creates atoms only within the specified region. If one desires the entire simulation box to be filled with atoms, one should never use the region-style with the region set to the entire simulation box. While it may seem equivalent to creating atoms with the box-style, LAMMPS then does not perform the extra checks that ensures exactly one atom is placed at a periodic boundary. This could otherwise happen due to numerical round-off errors.

To create a poly-crystalline model, the different atom species must be of different atom types. This can be achieved by selecting the lattice points for one atom type, creating atoms at those sites and then iterating for the number of atom types that is to be created. An alternative approach using a single custom lattice is to use the "basis" keyword to "create_atoms", specifying which basis atoms belong to which type.

Most actions in LAMMPS, such as setting velocities, forces, and which atoms that are written to a file, is done by referring to groups. The group "all" is automatically defined, and is dynamic - whenever and however atoms are added, "all" always refers to all atoms. User defined groups are static, and set up by assigning atoms of a certain type or region to a group. They can also be defined by using operations (such as "union", "intersect", and "subtract") on pre-existing groups.

Appendix A.5. Defining the potential

The type of interatomic potential used is set with "pair_style", and a potential file is read and mapped to the atom types (not species) with "pair_coeff". Force interactions are set for every atom pair (i,j), allowing different atoms in the model to have different interatomic potentials. If only one potential is used for all atoms, specify the atom pairs with a wildcard asterisk, (*,*), to set the same potential for all pairs at once. Potential files for many compounds are found in the NIST interatomic potentials repository at https://www.ctcms.nist.gov/potentials/. When building new potential files, refer to the documentation for the specific syntax for potential type used.

Many potential styles, such as Tersoff and Vashishta, allow the use of a "table" variant that can greatly reduce the total walltime (by a factor of 2-5). The speed up comes from that potential values at different distances are tabulated at the start of the simulation, instead of evaluated analytically throughout the simulation. (Linear interpolation between tabulation points is also performed). It is a trade-off between evaluation time and accuracy worth considering as the speed up is substantial even for a large amount of

tabulation points (roughly twice as fast for InSb with one million tabulation points).

The mass for each atom type is set with "mass". It is required to be set for all potential types except eam (where it is already specified in the potential file).

Appendix A.6. Building the model

Velocities are set with "velocity". The velocities are set once when this command is encountered, not continuously throughout the run. Creating randomized velocities for the entire system at a certain temperature is done with the "create" keyword.

Interactions that apply continuously throughout the run are set with "fixes". "fix setforce" manually sets forces on a group of atoms. This is useful for holding atoms in place, at a constant velocity, or to linearly accelerate them. "fix nvt" specifies a canonical ensemble, and "fix nve" a microcanonical ensemble. Using several fixes at once, such as "setforce" and "nve" is possible.

"Computes" define additional quantities that are to be calculated throughout the run. This can be system-quantities as well as per atom quantities. Computes are referenced similarly to variables but with the "v" replaced with a "c", for instance "c_myUserGivenComputeName". For computes with several output quantities per atom, for instance "compute stress/atom", the i:th vector is referenced with "c_stressCompute[i]". Specific elements can be referenced with "c_stressCompute[i][j]". Useful per atom computes include kinetic energy ("ke/atom"), potential energy ("pe/atom"), stress tensor times volume ("stress/atom"), and Voronoi volume ("voronoi/atom"). Atom properties that are always calculated to perform the time-integration, such as positions and velocities, can be accessed through "compute property/atom".

Appendix A.7. Generating outputs and binning

Variable and compute values are not accessible once the run is completed unless they are specifically written to a file. A string with variables evaluated with "${}" can be written to a file with "fix print". Atom properties and computes can be dumped to a file with "dump". The dump-type must be "cfg" and at least include the properties "mass type xs ys zs id" for OVITO to read the dump file later on ("xs" "ys" and "zs" denote atom coordinates scaled between 0 and 1). The wildcard "*" should be included in the filename if several dumps are made, to prevent files from overwriting each other. The

wildcard is replaced by the current timestep when the file is written, guaranteeing unique filenames.

LAMMPS can output data to the terminal window throughout the simulation. This does not work on Aurora (although everything that is written to the terminal window is also written to the default output file "log.lammps", which is still accessible). "thermo" is used to determine how often LAMMPS writes this output, and "thermo_style" is used to determine what quantities are written.

To output data averaged over groups of atoms, or even time averaged data, one can use binning. "compute chunk/atom" subdivides a group of atoms into chunks that can be averaged over. Then "fix ave/chunk" performs the averaging, and outputs the result to a file if the "file" keyword is appended. (Old LAMMPS-code examples will use the "fix ave/spatial" command for this purpose, however this command is not available in LAMMPS since the 2015 versions).

Appendix A.8. Running the simulation

The simulation is run for N time-steps with "run N". This takes place when this command is encountered, so all relevant fixes and computes must have been defined beforehand. It is possible to have successive "run" commands, with newly defined computes and fixes between each run. Deletion of old fixes and computes are done with "uncompute" and "unfix". The timestep can also be reset to start at zero for the next run with "reset_timestep 0".

It is further possible to run an unknown amount of timesteps with "minimize". This command iterates atom positions until the energy of the system is minimized (within user given stopping criteria).

Appendix B. Introduction to OVITO

Visualization of LAMMPS-simulations and computations of kinematic and structural quantities can be done in the open visualization tool OVITO. OVITO can be downloaded from https://www.ovito.org/. The full documentation is found at https://www.ovito.org/docs/current/.

Appendix B.1. Basic operations and general workflow

To load in atoms from cfg-files, click "load file" and select the first dump-file. If subsequent files have the same filename except for an increasing integer, OVITO automatically loads them as well. A slidebar, used to move

between frames, is shown at the bottom left. Camera controls are found to the right of the slidebar. To play an animation of the frames back to back, simply press the play-button next to the slidebar. This function unfortunately does not work well for larger systems, as the load time to move between frames becomes too large (around 10 seconds for 4 million atoms on Aurora, several minutes on my local laptop).

Operations such as computing properties, moving and colouring atoms, are all done by selecting "Add modification". Selecting a modification adds it to the "Modifications"-list to the right of the viewports. Modifications are executed from the bottom up. Move modifiers up and down with the arrows on the right, and un-tick the box to the left of a modifier to temporarily deactivate it. Output images are generated in the "Rendering" tab with a camera on the right of the "Pipelines" tab. Colourbars and text can be added to the figures in the rightmost "Viewport layers" tab before rendering.

The entire session state (including loaded atoms and applied modifiers) can be saved under "File - Save session state". However as the number of used modifiers can be large, and they appear with generic default names such as "Compute Property" in the modifications-list, it can be hard to quickly get an overview of what a previous sessions state was doing when reloading it later on.

A more convenient way to work with OVITO is to save groups of modifiers that are always used together as a "Modifier Template". They can then later be added together as a single modification in the next OVITO session. This allows a model to be built quickly from scratch each time, with only the relevant parts for the current post-processing present in the modifications list. Modifier templates are created by pressing "Manage Modifier Templates" on the right of the modifications list.

Appendix B.2. Concrete example of generating figures

This subsection describes in detail how to create figures 6 and 7, in order to illustrate how to work with OVITO and reproduce all figures in this work.

Load the cfg-files containing the atoms. Use the slidebar at the bottom left to move to the frame corresponding to the time you want to study. Once loaded all atoms appear grey. To change this, add the "Color coding" modifier from the drop down menu to the right. Once added, select it in the modifications list. Details on the modifier is now displayed below the modifications list. Selecting "Input property", a list of all available per-atom quantities in the dumped file is shown. Selecting to colour by "Mass" and

adjusting the colourbar by pressing "Adjust range", figures like figure 15 and 16 can be created in InSb.

To create figure 6 we want to colour by coordination number, a property that first needs to be calculated in OVITO. Add the "Coordination analysis" modifier. Move it down in the modifications list so it executes before the colouring. Select a cutoff radius of 10 Å with 1000 histogram bins. A progress bar of the calculation is shown below the slidebar. Once the calculation is finished (after roughly 15 seconds), a plot of the RDF is shown. (This data can be exported to a text file by pressing "Show in data inspector", "Data table view", and "Export data to text file". A script to read such files and calculate the XRD from them is found in appendix E.1). The input property "Coordination" should now be available to the "Color coding" modifier. Selecting it the the shockwave and the amorphous bands in germanium should be visible by coordination number alone.

To investigate the shocked material one can run structure identification algorithms. Temporarily disable the two active modifications and add the "Identify diamond structure" modifier. Atoms are now coloured by the structure of their local environment. The crystalline atoms above the shockfront are identified as diamond cubic, while most atoms below the shockfront are unidentified (denoted "Other"). This is as the material below the shockfront is compressed, something the algorithm does not consider. Add the "Affine transformation" modifier before the structure identification. Define the transformation such that the simulation box is stretched to its initial size. The transform can be defined by explicitly stating a transformation matrix, in which case one needs to calculate by hand the appropriate scaling factor in the compression direction. Alternatively the transformation can be given by stating the desired simulation box dimensions, which can be read from the LAMMPS log-file. Once stretched, only the bands should be unidentified. Add the inverse "Affine transformation" modifier after the structure identification algorithm to return the system to its initial size.

The affine transformation does not have to completely un-strain the lattice for the structure identification to work. (In fact, stretching the simulation box to its initial size the atoms below the shockfront are still slightly compressed and the atoms above the shockfront are stretched). For a diamond cubic lattice at temperature 300 K, re-scaling along [001] with factors between 0.85 to 1.30 still leads to a correctly identified lattice. Additionally, the algorithm considers periodic boundaries.

To recreate figure 6, we now want to colour only the diamond cubic

atoms by coordination again. Select these atoms by adding the "Select type" modifier. Chose property "Structure type" and tick the boxes for "Cubic diamond" and "Cubic diamond (1st neighbor)". The selected atoms now appear red. Re-enable the "Color coding" and "Coordination analysis" and place them at the top of the modifications list. In the "Color coding" modifier, tick the box "Color only selected elements". Figure 6 should now have been recreated.

To show only the amorphous bands, disable the "Color coding" and "Coordination analysis" modifiers and replace them with the "Delete selected" modifier. Only the bands are now shown, but without any indication of depth. Add the "Ambient occlusion" modifier and the system should look like figure 7. To examine the bands in more detail, the "Slice" modifier can give a cross section or slice out a slab of given thickness.

Two of the most versatile modifiers are "Compute property" and "Select expression". "Compute property" enables calculation of arbitrary properties, such as per-atom stress tensors or the von Mises stress, with user given expressions. Both properties in the dump file and properties calculated by a previous "Compute property" modifier can be used. "Select expression" selects atoms based on a user given boolean expression.

Properties that are output for some but not all atoms in the simulation default to a value of zero for the atoms with no information in the dump file. In the LAMMPS script, properties such as stress are not dumped for the piston. This can be used to select the piston, for instance in order to remove it before running coordination analysis. Since the piston is frozen throughout the simulation, including it in the RDF leads to undesirable spikes in the histogram at the perfect lattice sites. It should therefore be removed if the RDF is to be used to calculate the XRD.

Finally, remember to save the active modifications as a "Modifier template" if they are to be used together again on other data.

Appendix C. Utilizing the LU cluster Aurora

This section is intended as a guide on how to start using the LU computer cluster Aurora, aimed towards newcomers to clusters and parallel computing. For more in-depth descriptions the reader is referred to the LUNARC-documentation, https://lunarc-documentation.readthedocs.io.

Appendix C.1. Accessing Aurora

It is possible to access Aurora completely through the command line, however it is recommended to download additional clients to interact with Aurora through a graphical desktop. (This is required to run Graphical User Interface-programs like OVITO on Aurora). The graphical desktop is reached through ThinLinc, set-up instructions are found in the LUNARC documentation at:
https://lunarc-documentation.readthedocs.io/en/latest/using_hpc_desktop/. Logging in a regular Linux-desktop is shown, that can be interacted with as on a local computer (e.g. create folders, move files, and even write scripts in a text editor).

Transferring files, such as LAMMPS input scripts, potential files, and results, to and from Aurora can not be done with drag and drop from a local desktop onto the graphical ThinLinc-desktop. For this, Windows users can download an additional client called MobaXterm. Instructions on how to set it up is found at: https://www.lunarc.lu.se/support/tutorials/tutorial-login-using-mobaxterm-windows/ (written by the LUNARC-team, but located on a site outside the regular LUNARC documentation).

Appendix C.2. Aurora structure and specifications

A physical computer is referred to as a "node". Each node has several processing elements, known as "cores", as well as its own memory and a local disk. There are 20 cores per node, and in total 62 GB memory to use per node, giving each core 3.1 GB per default.

Aurora consists of hundreds of interlinked nodes, with 180 available for SNIC-use. A few nodes are dedicated to handling user home-spaces, and submitting jobs. These nodes are referred to as the "front-end". The rest of the nodes, referred to as the "back-end", are for performing computations.

Upon log-in to Aurora, one is placed at the front-end. In the home-space "/home/username" one has access to 100 GB of storage (temporary overdrafts of up to 150 GB is allowed). All files are regularly backed up. If more storage space is required, an application for a SNIC storage project can be made.

As the front-end nodes are shared between hundreds of users, it is strictly forbidden to run any compute intense process here (such as executing LAMMPS form the command line). To run computations, the process must first be sent to the back-end. This is done through a job-script.

Appendix C.3. Job-script structure

The job-script is handled by the batch system SLURM, with some minor Aurora specific modifications. The purpose of the job-script is to transfer instructions on what is to be computed to a suitable number of cores. All job scripts contain:
- Shell specification.
- Resource specification.
- UNIX-script.

The job-script is read as a regular UNIX-script, meaning comments can be made with "#". The exception is the shell specification, and commands regarding resource specification. The latter all start with "#SBATCH". The resources one must specify is the number of nodes, cores per node, and maximum walltime. These statements are typically changed between runs. One can also allocate more memory per core, though this will require using fewer cores per node.

The maximum walltime refers to real-time (not core-h). If the job is still running when the maximum walltime is reached, the job is terminated. This acts as a safety mechanism preventing one from burning one's allocation on a single hanging job, but it also means that the walltime has to be over-specified.

It is recommended to request exclusive access to all nodes used, as otherwise SLURM might place other users on the same node if there are more cores available. This could potentially lead to errors, so typically one will use multiples of 20 cores at all times to fully utilize the requested nodes.

To run installed software on Aurora, the software must first be loaded onto the nodes on the back-end. This is done through "modules". Executing "module spider" in the Aurora command line shows all modules that can be loaded, LAMMPS is in this list. Executing "module spider moduleName" shows more information about the module "moduleName", including the name of other modules that possibly has to be loaded beforehand. Once all modules are loaded, LAMMPS can be run with commands from the LAMMPS-input script.

A conventional extension for the job-script file is ".sh". The job-script is run from the command line with the command "sbatch myJobScript.sh". SLURM should then reply by giving the job a unique job-id. The job-script is run from the same directory as the LAMMPS-input script. Output files are by default written to the same directory.

An example job-script that runs LAMMPS is found in appendix E.2.

Appendix C.4. Workflow and general tips

Customize the LAMMPS input script and run small tests on a local computer. Upload the script to Aurora with MobaXterm. Log onto the virtual desktop through ThinLinc, and modify the resource specification in the job-script. Submit with "sbatch" in the terminal. Results is later downloaded using the MobaXterm client. OVITO on Aurora is accessed from the ThinLinc desktop. In the drop-down menu, select "Applications - LUNARC Applications - Post Processing - OVITO". This starts OVITO on a back-end node (without using time on the allocation).

To monitor the job-status, write "squeue -u userName". This shows all jobs currently submitted for the user "userName". "PD" means it is pending, i.e. waiting for resources to start, "R" that it is running and "CG" that it is completing. A job is manually cancelled by submitting "scancel job-id". Specifying few nodes and a short maximum walltime means that the job is more likely to start faster, as the batch system does not start jobs in the order they are submitted but instead attempts to maximize usage of all Aurora nodes. Using a maximum walltime of 5-10% higher than the anticipated one is a good rule of thumb (the actual walltime will vary slightly between identical runs). The queue also prioritizes users that have only used a small amount of their allocation.

Always run smaller tests to get reasonable walltime estimates. Knowing that the MD walltime is roughly proportional to the number of atoms (slightly worse) is enough to get reasonable estimates.

The allocation is measured in core-h, which is the real time a job takes times the number of cores used. Executing "projinfo" in the command line shows how many core-h that has been used thus far for a given month. In the supr-protal under "projects", one can also see this illustrated with histograms showing used core-h day by day, and user by user. (Note that these are only updated once per day). Executing "snicquota" in the command line shows how much space is used on the home-space. This space can potentially fill up rapidly. The dominating contributor is the OVITO cfg-dump files. A dump with 4 million atoms can take up around half a GB, meaning that generating a movie with 35 frames requires outputs of around 20 GB (a fifth of the total quota). Be selective with the amount of dump files written, often only a handful of frames is enough to determine if the simulation was successful or can be discarded.

Several jobs can run simultaneously. Make sure the output files do not overwrite each other, by all having different names of by writing to different directories.

If further help concerning Aurora is needed, a support request to the LUNARC-team can be issued from the supr-portal.

Appendix D. Benchmarks

This section details the walltimes for some typical production runs.

Compressing germanium along [001], with a piston speed of 1.2 km/s, 1 fs timestep, 10 ps equilibration, and 8.75 ps shock sequence, took 33 minutes and 9 seconds on ten nodes. This corresponds to 110.5 core-h. This was without invoking the table-style of the Tersoff-potential in LAMMPS. Invoking the table-style and rerunning the simulation with five nodes took 70.53 core-h, and another rerun with three nodes took 70.32 core-h. While running on fewer processors should lower the used core-h, the difference of 0.21 core-h is also within the typical walltime variation successive runs with equal conditions has. No run in germanium utilized processor rebalancing.

Compressing InSb along [111], with a piston speed of 1.2 km/s, 1 fs timestep, 10 ps equilibration, and 10.92 ps shock sequence, on ten nodes, took 533.3 core-h. This was without invoking the Vashishta potentials table-style in LAMMPS, and without any processor rebalancing. A later simulation with piston speed 1.1 km/s, identical timestep and equilibration time, 12 ps shock sequence (slightly longer) on five nodes took 230.1 core-h.

The reduced computation time was achieved with processor rebalances in [111] every 0.5 ps, and invoking the Vashishta table-style with a million tabulation points. These settings were also used for the largest simulation, that used a timestep of 2 fs, 10 ps equilibration, and 50 ps shock sequence in total. This simulation took 1036 core-h on five nodes.

Compression in InSb along [111] with non-periodic boundary conditions with a piston speed of 1.15 km/s, using a timestep of 2 fs, 4 ps equilibration, and 12 ps shock sequence took 95.06 core-h on five nodes. This run had processor rebalancing every 2 ps. The processors were not only rebalanced along [111], but in all dimensions using the recursive bi-sectioning algorithm implemented in LAMMPS. This balances the particles evenly across all processors while allowing for processor sub-domains that are not strictly on the default rectilinear grid. Processors with very few to no atoms in the empty

space of the simulation domain above the material expanding normal to [111] in figure 24 is thus avoided.

The equilibration typically runs faster than the shock sequence. This is likely due to that the atoms are less evenly distributed to the processors during the shock sequence, which can be partly mitigated through processor rebalancing.

Appendix E. Used scripts

Appendix E.1. MATLAB script for XRD from RDF

```
% Plots XRD-pattern, S(q), from RDF, g(r). Calculation done with a truncated sine-transform and a centered sine-window.
% The calculation assumes equal scattering from all atoms in the lattice, meaning 200, 222, 420... peaks will not appear for a zinc-blende lattice (as in a diamond cubic lattice). This is a good approximation for InSb as In and Sb have similar atomic scattering factors, though it might be a problem if other materials (such as InP) is studied.
% User given input variables:
filename = 'file_with_RDF';
R = 80; % Cutoff in Å, units and truncation must agree RDF-file
qmin = 1; % Minimum plotted scattering vector in Å^-1.
qmax = 7; % Maximum plotted scattering vector in Å^-1.
Nq = 10000; % Number of examined scattering vectors.

% Reads file
struct_gr = importdata(filename);
gr = struct_gr.data;
r = gr(:,1);
g = gr(:,2);

% Sets up vectors and window-function for the integration
q = linspace(qmin,qmax,Nq)'; % Scattering vectors examined.
S = zeros(1,length(q))'; % Scattered intensity
w = cos(pi.*(r-0.5.*R)./R); % Centered sine-window

% Numerical integration (truncated sine-transform)
for i = 1:length(q)
```

```
    S(i) = 1+trapz(r , w.*g.*sin(q(i).*r))./q(i);
end

% Plotting
plot(q,S);
grid on;
xlabel('Reciprocal position q [Åˆ{-1}]');
ylabel('Scattered intensity S [a.u.]');
title('XRD');
hold on;
```

Appendix E.2. Aurora job-script example

```
#!/bin/sh
# Shell specification.

# Requests 5 nodes, with 20 cores on each.
#SBATCH -N 5
#SBATCH –tasks-per-node=20

# Gives exclusive access to the requested nodes.
#SBATCH –exclusive

# Requests 3100 MB memory per core. This is the default request.
#SBATCH –mem-per-cpu=3100

# Maximum walltime in hours:minutes:seconds.
#SBATCH -t 00:13:37

# Prevents job from automatically re-queing if it crashes.
#SBATCH –no-requeue

# Files for output and errors of the job-script. %j includes the job-id in the
filename, guaranteeing that output files never overwrite each other. Identical
filenames means that standard output and errors are written to the same file.
#SBATCH -o LAMMPSOutput_%j.out
#SBATCH -e LAMMPSOutput_%j.out
```

```
# Write this script to output-file
cat $0

# Name of job in queue. Max 8 characters.
#SBATCH -J goodJob

# Deletes previously loaded modules.
module purge

# Load LAMMPS
module load GCC/9.3.0
module load OpenMPI/4.0.3
module load LAMMPS/3Mar2020-Python-3.8.2-kokkos

# Run the job. Executable is "lmp".
mpirun -bind-to core lmp -in LAMMPS_input_script.in

echo "Job-script done!"
```

Appendix E.3. LAMMPS input script

```
# Shock Script Modified to compression along [111] in InSb and different
piston behaviour
# Based on a script by Oscar Guerrero-Miramontes
# Updated to LAMMPS 2021 and tweaked by Erik Löfquist

#———Initialize Simulation ————

clear
units metal
dimension 3
boundary p p p #Changes after equilibration to s along z.
atom_style atomic
atom_modify map array

#———Define variables————————
```

```
# Controlling processors and balancing.
variable rebalanceEvery equal 0.5 #Time in ps between processor
rebalances. Each takes around 0.02 s (real time)
# processors 5 4 5 # Manual mapping of processors. If used, make sure the
product matches the nbr of currently used nodes.

# Material characteristics
variable alattice equal 6.479 # lattice constant (unit Å)
variable massIn equal 114.8 # mass in u
variable massSb equal 121.8 # mass in u

# Simulation parameters
variable stemperature equal 300 # temperature in K
variable myseed equal 12345 # RNG velocity-seed
variable vpiston equal 1.2 #1.2 # max piston speed in (km/s), multiply by
ten to get (Å/ps)

# Dimensions
variable xmax equal 44 # size in x-direction (units lattice)
variable ymax equal 38 # size in y-direction
variable zmax equal 53 # size in z-direction

# Simulation times
variable time_step equal 0.001 # timestep in picoseconds
variable t_eq equal 10 # equilibration time in ps
variable t_shock equal 14 # time in ps for entire simulation (excluding equi-
libration)
variable tpiston_accel equal 1 # duration of piston acceleration in ps
variable tpiston_compress equal 5 # duration of piston moving with constant
velocity
variable tpiston_decel equal 2 # duration of piston deceleration in ps
# Always manually check that t_shock >= tpiston_accel + tpiston_compress
+ tpiston_decel

# Control behaviour of output files
variable Nevery equal 10 # use input values every this many timesteps
variable Nrepeat equal 5 # number of times to use input values for calculat-
ing
```

```
variable Nfreq equal 100 # calculate averages every this many timesteps
variable Nthermo equal 100 # Print to command line after Nthermo timesteps
variable deltaz equal 3 # thickness of spatial bins in Å.
variable atomrate equal 1000 # the rate in timestep that atoms are dumped
as CFG

#————Intermidiate Calculations. Not modified between runs————

# Set damping parameters for the thermostatting and barostatting during
nvt equilibration.
# Follows recommendation of 100 and 1000 timesteps given in the documen-
tation.
variable tdamp equal "v_time_step*100"
variable pdamp equal "v_time_step*1000"

variable Up equal "10*v_vpiston" #Unit conversion to Å/ps

# Calculates constant acceleration force for piston in eV/Å
variable forceZperMass_accel equal "1.036426958*0.001*v_vpiston/v_tpiston_accel"
variable forceZIn_accel equal "v_forceZperMass_accel*v_massIn"
variable forceZSb_accel equal "v_forceZperMass_accel*v_massSb"

#Corresponding for the deceleration of the piston (Note sign)
variable forceZperMass_decel equal "1.036426958*0.001*v_vpiston/v_tpiston_decel"
variable forceZIn_decel equal "-v_forceZperMass_decel*v_massIn"
variable forceZSb_decel equal "-v_forceZperMass_decel*v_massSb"

# Converting time in ps to nbr of timesteps. MUST BE INTEGERS!
variable time_eq equal "v_t_eq/v_time_step"
variable time_shock equal "v_t_shock/v_time_step"
variable time_accel equal "v_tpiston_accel/v_time_step"
variable time_compress equal "v_tpiston_compress/v_time_step"
variable time_decel equal "v_tpiston_decel/v_time_step"
variable time_remaning equal "v_time_shock - v_time_accel - v_time_compress
- v_time_decel"
variable Nrebalance equal "v_rebalanceEvery/v_time_step"

# Sets the timestep
```

```
timestep ${time_step}

#————Create Atoms & Divide into groups————————

# Creates Zinc-Blende lattice.
variable originShift equal "sqrt(3)/4" #Shifts lattice to create whole layers in (111)
lattice custom ${alattice} &
a1 1.0 0.0 0.0 &
a2 0.0 1.0 0.0 &
a3 0.0 0.0 1.0 &
basis 0.0 0.0 0.0 basis 0.5 0.0 0.5 basis 0.0 0.5 0.5 basis 0.5 0.5 0.0 &
basis 0.25 0.25 0.25 basis 0.75 0.25 0.75 basis 0.25 0.75 0.75 basis 0.75 0.75 0.25 &
origin 0.0 0.0 ${originShift} orient x 1 0 -1 orient y -1 2 -1 orient z 1 1 1
# For compression in [001] instead of [111], change last line to:
# origin 0.0 0.0 0.0 orient x 1 0 0 orient y 0 1 0 orient z 0 0 1

# Set up simulation box
region sim_box block 0 ${xmax} 0 ${ymax} 0 ${zmax} units lattice
create_box 4 sim_box

create_atoms 2 box basis 1 2 basis 2 2 basis 3 2 basis 4 2 &
basis 5 1 basis 6 1 basis 7 1 basis 8 1

# Assign groups and types. 1=InPiston 2=SbPiston, 3=InBulk, 4=SbBulk
region pistonRegion block INF INF INF INF INF 4
region bulkRegion block INF INF INF INF 4 INF

group piston region pistonRegion
group bulk region bulkRegion
group InAtoms type 1
group SbAtoms type 2

group InPiston intersect InAtoms piston
group SbPiston intersect SbAtoms piston
group InBulk intersect InAtoms bulk
group SbBulk intersect SbAtoms bulk
```

```
set group InPiston type 1
set group SbPiston type 2
set group InBulk type 3
set group SbBulk type 4

#————————Define Interatomic Potential & Computes————————

pair_style vashishta/table 1000000 0.2
pair_coeff * * InSb2.vashishta In Sb In Sb

mass 1 ${massIn}
mass 2 ${massSb}
mass 3 ${massIn}
mass 4 ${massSb}

compute myCN bulk cna/atom 1 #Centro-symmetry (useless for InSb)
compute myKE bulk ke/atom #Kinetic energy
compute myPE bulk pe/atom #Potential energy
compute myCOM bulk com #Center of mass
compute peratom bulk stress/atom NULL # Stress times volume tensor
compute vz bulk property/atom vz #Velocity in z

#Debug check of piston velocity
compute vzPiston piston property/atom vz

# Gives voronoi-volume and coordination number for each atom. Useful for
calculating stresses later.
compute myVoronoi all voronoi/atom

#————————Equilibrate————————————-
reset_timestep 0

# Assigns initial velocities using a Maxwell-Boltzmann distribution & Equi-
librates with nvt ensemble.
velocity bulk create ${stemperature} ${myseed} rot yes dist gaussian fix
equilibration bulk npt temp ${stemperature} ${stemperature} ${tdamp} &
iso 0 0 ${pdamp} drag 1
```

```
# Prints equilibration characteristics to file
variable eq1 equal "step"
variable eq2 equal "pxx/10000" #(Converts pressure in bar to GPa)
variable eq3 equal "pyy/10000"
variable eq4 equal "pzz/10000"
variable eq5 equal "lx" #Length of simulation box in x
variable eq6 equal "ly"
variable eq7 equal "lz"
variable eq8 equal "temp" #System temperature
variable eq9 equal "etotal" #System energy
fix output1 bulk print ${Nthermo} "${eq1} ${eq2} ${eq3} ${eq4} ${eq5}
${eq6} ${eq7} ${eq8} ${eq9}" &
file initialConditions.out screen no

# Prints to command-window (or standard job-output file on Aurora)
thermo ${Nthermo}
thermo_style custom step pxx pyy pzz lx ly lz temp etotal

run ${time_eq}

unfix equilibration
unfix output1

print "Equilibration complete, initializing shock calculations..."

#————————-Sets-up shock sequence————————————————

# Lifts periodic boundary in z change_box all boundary p p s
reset_timestep 0

# Creates bins to track the passing of the shockwave. Outputs each profile
to a file.

compute chunkCompute bulk chunk/atom bin/1d z lower ${deltaz} units box

fix density_profile bulk ave/chunk ${Nevery} ${Nrepeat} ${Nfreq} & chunkCom-
pute density/mass file denz.profile
```

```
variable temp atom c_myKE/(1.5*8.61e-5)
fix temp_profile bulk ave/chunk ${Nevery} ${Nrepeat} ${Nfreq} & chunkCompute v_temp file temp.profile

#Temperature neglecting COM movement for each bin.
compute tempChunkCompute all temp/chunk chunkCompute temp com yes
fix temp_profile_corrected all ave/time ${Nevery} ${Nrepeat} ${Nfreq} & c_tempChunkCompute[1] file temp_COMcorrected.profile mode vector

variable meanpress atom -(c_peratom[1]+c_peratom[2]+c_peratom[3])/3
fix pressure_profile bulk ave/chunk ${Nevery} ${Nrepeat} ${Nfreq} & chunkCompute v_meanpress file pressure.profile

fix vel_profile bulk ave/chunk ${Nevery} ${Nrepeat} ${Nfreq} & chunkCompute c_vz file velocityZcomp.profile

#Extra fix to check if piston behaves correctly
compute chunkComputePiston piston chunk/atom bin/1d z lower ${deltaz}
units box fix piston_vel_profile piston ave/chunk ${Nevery} ${Nrepeat} ${Nfreq}
&
chunkComputePiston c_vzPiston file velocityZpiston.profile

#Prints system characteristics to file
variable eq1 equal "step"
variable eq2 equal "pxx/10000"
variable eq3 equal "pyy/10000"
variable eq4 equal "pzz/10000"
variable eq5 equal "lx"
variable eq6 equal "ly"
variable eq7 equal "lz"
variable eq8 equal "temp"
variable eq9 equal "etotal"
variable eq10 equal "c_myCOM[3]"
fix shock bulk print ${Nthermo} "${eq1} ${eq2} ${eq3} ${eq4} ${eq5}
${eq6} ${eq7} ${eq8} ${eq9} ${eq10}" &
file run.${stemperature}K.averagedQuantities.out screen no
```

```
# Command window output
thermo ${Nthermo}
thermo_style custom step pxx pyy pzz lx ly lz temp etotal c_myCOM[3]

# Dumps cfg-files for OVITO post-processing
dump OVITOdump all cfg ${atomrate} dump._*.cfg mass type xs ys zs id
c_myPE c_myKE c_myCN &
c_peratom[1] c_peratom[2] c_peratom[3] c_peratom[4] c_peratom[5] c_peratom[6]
&
c_myVoronoi[1] c_myVoronoi[2]

# Extra dump of atom positions. Can be used with fix xrd in another run
to obtain virtual xrd-spectra. (Not used)
# dump xrdDump bulk custom ${atomrate} InSbxrd.dump id type x y z

# Rebalancing of processors.
fix rebalancingFix all balance ${Nrebalance} 0.9 shift z 20 1.0001

#———-Running the shock sequence———————

# Linearly accelerates piston to velocity Up
velocity piston set 0 0 0 sum no units box
fix nveAcc all nve
fix InAcc InPiston setforce 0.0 0.0 v_forceZIn_accel
fix SbAcc SbPiston setforce 0.0 0.0 v_forceZSb_accel

run ${time_accel}

unfix nveAcc
unfix InAcc
unfix SbAcc

# Compress with constant piston velocity Up
fix nveCompress all nve
velocity piston set 0 0 v_Up sum no units box #Should be redundant.
fix pistonLin piston setforce 0.0 0.0 0.0

run ${time_compress}
```

```
unfix nveCompress
unfix pistonLin

# Piston deceleration to rest

fix nveDecel all nve
fix InDecel InPiston setforce 0.0 0.0 v_forceZIn_decel
fix SbDecel SbPiston setforce 0.0 0.0 v_forceZSb_decel

run ${time_decel}

unfix nveDecel
unfix InDecel
unfix SbDecel

# Run remaning time with fixed piston.

fix nveRemaning all nve
velocity piston set 0 0 0 sum no units box #Should be redundant.
fix pistonFreeze piston setforce 0.0 0.0 0.0

run ${time_remaning}

print "All done! Did not crash or hit walltime limit."
```