Tampere University

Eerik Voimanen

MOLECULAR DYNAMICS SIMULATIONS OF NANOPILLAR COMPRESSION

Master of Science Thesis
Faculty of Engineering and Natural Sciences
Examiners: Postdoctoral Research Fellow Marcin Mińkowski
Professor Lasse Laurson
January 2023

ABSTRACT

Eerik Voimanen: Molecular Dynamics Simulations of nanopillar compression
Master of Science Thesis
Tampere University
Master's Programme in Science and Engineering
January 2023

Crystal plasticity experiments consider fundamental physics of complex systems. With simulations, statistical analysis is performed on nanopillar compression. With the statistical data this thesis has done research for better understanding of the structure to yield stress relation.

This thesis considers tantalum polycrystal nanopillar compression with two datasets. First of the two datasets considers 20 nm midway diameter Ta nanopillars with different grain sizes and indenter velocities. From the research done with the first dataset, there seems to be two regimes for the yield stress dependency on the grain size. For the average grain size larger than roughly 35 nm there seems to be a high dependency on the yield stress if the grain size is modified. If the grain size is increased, the yield stress increases, approaching the monocrystal yield stress. For grain sizes smaller than that, there seems to be almost no effect on the yield stress when it comes to modifying the grain size.

With statistical analysis on another dataset, which considers 20 nm to 50 nm midway diameter Ta nanopillars, it is also proven that statistical analysis is necessary to draw reliable conclusions. This is proven by a fact that a similar study, with some differences, but still more than enough in common with this research, has different conclusions with single samples than with this study has with a statistical analysis.

Keywords: molecular dynamics, Nanocrystal generator, yield stress, statistics, polycrystal, tantalum

TIIVISTELMÄ

Eerik Voimanen: Molekyylidynamiikkasimulaatiot nanopilareiden puristuksesta
Diplomityö
Tampereen yliopisto
Teknis-luonnontieteellinen DI-ohjelma
Tammikuu 2023

Kiteisten aineiden plastisuuskokeet käsittelevät monimutkaisten järjestelmien fundamentaalia fysiikkaa. Simulaatioiden avulla on mahdollista tehdä tilastollista analyysiä nanopilarien puristuksesta. Tässä työssä statistista dataa käytetään hankkimaan lisätietoja aineen rakenteen ja myötörajan suhteesta.

Työssä on tuotettu kaksi tietojoukkoa monikiteisten tantaalinanopilarien puristuksesta. Ensimmäinen kahdesta tietojoukosta koostuu 20 nm keskikorkeuden halkaisijan katkaistun kartion muotoisista Ta nanopilareista. Tutkimuksessa huomattiin kaksi aluetta raekoon ja myötörajan suhteesta. Jos keskimääräinen raekoko on suurempi kuin 35 nm, raekoon muuttaminen vaikuttaa voimakkaasti myötörajan jännityksen arvoon. Jos raekokoa kasvatetaan, myös myötörajan jännitys kasvaa lähestyen yksikiteisen pilarin myötörajan jännitystä. Tuota raekokoa pienemmille raekoolla ei ole juurikaan vaikutusta myötörajan jännityksen arvoon.

Toisen tietojoukon, joka koostuu pilareista 20 nm – 50 nm keskikorkeuden halkaisijoilla, perusteella todistetaan, että johtopäätösten tekeminen vaatii tilastollista analyysiä. Tämä on todistettu vertailemalla tuota dataa toiseen tästä tutkimuksesta poikkeavaan, mutta silti tarpeeksi samankaltaiseen, tutkimukseen. Tuo tutkimus on päätynyt yksittäisten tulosten takia eri johtopäätökseen kuin tämä työ tilastollisella analyysillä.

Avainsanat: molekyylidynamiikka, Nanocrystal generator, myötöraja, tilasto, monikide, tantaali

PREFACE

Not all the theses are created in a perfect fantasy world where everything always goes according to plans. Not all the data is always valid, and not all the results are always groundbreaking. But sometimes, rather always, there is a point that can be learnt from anything. For this thesis it might have been the fact that data invalidation does not mean that nothing was learnt and all time went to waste. The journey, the joy and the terror that has led to this point has taught something. Something, a skill to create statistically sound data, and to compare the statistical data to already existing data. Let statistics never fail to exist, since it is impossible to escape from.

I would also like to thank everyone who have supported me in my journey to finishing this thesis, including but not limited to my family, my supervisors Postdoctoral Research Fellow Marcin Mińkowski and Professor Lasse Laurson, my colleagues and friends.

I also wish to acknowledge CSC – IT Center for Science, Finland, for generous computational resources. Also thank you for Dr. Carlos Ruestes from Universidad Nacional de Cuyo for sharing the potential file for the simulations.

Tampere, 16th January 2023

Eerik Voimanen

CONTENTS

ACRONYMS

BCC	Body centered cubic
Efs	Extended Finnis-Sinclair potential
FCC	Face centered cubic
LAMMPS	Large-scale Atomic/Molercular Massively Parallel Simulator [1]
OVITO	Open visualization tool[2]
SC	Simple cubic
SD	Standard deviation
SEM	Standard error of mean

1. INTRODUCTION

One of the most important aims in material science is to study and establish the relation between the structure and the properties of the material [3, 4, 5]. Pillar compression experiments and simulations are possible methods of gathering the mechanical response data [6, 7, 8, 9]. In such experiments the mechanical response is measured during deformation. Traditionally, the larger macro-scale material pieces have been compressed and the mechanical response has been measured during the compression. It is also possible to experiment with smaller pillars manufactured with carving methods such as focused ion beam. Experiments on crystal plasticity have been performed with different size scales. A few different size effects have been discovered with those different sized compression experiments, one of them is the fact that smaller pillars seem to be stronger [10]. The other size effect is smaller pillars having more noisy stress-strain curves [11]. This comes from the fact that a certain absolute indenting plane movement affects the stress and strain of a smaller pillar more than those of a larger pillar.

The research done for this thesis aims to give a statistical approach to the relation between the structure of the nanopillar and its mechanical properties. For the statistical data, several different initial grain configurations have been used to mitigate the effects of random variation.The results are also compared to a rather similar study which uses, however, a slightly different methodology [8].That study has used single polycrystals as the samples, which makes the approach not statistical. There are also other differences such as a different ensemble, timestep and a different method for generating the samples. If differences between those two studies are to be found, such as discrepancy between the relations of grain size and yield stress or different conclusion, they are explained by the different approach to statistical data in addition to differences in programs, ensembles and other settings used.

In chapter 2 the structure of crystalline materials is introduced together with Voronoi tessellation and other theoretical concepts related to pillar compression experiments and simulations. Chapter 3 introduces a new tool for creating the pillars for the simulations and compares the current feature set to a more widely used program Atomsk [12]. Chapter 4 includes more information about molecular dynamics simulations, both in general and some specific information related to simulations done in this thesis. In chapter 5 the analysis software is introduced, and after that the results of the simulations are shown. In

that chapter it is shown that the results obtained for one of the datasets contradict those of an analogous study found in the literature [8]. The chapter 6 ends the thesis with the conclusions.

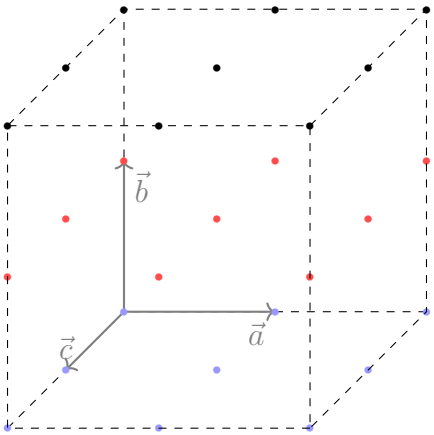
2. THEORY AND BACKGROUND

2.1 Structure of crystalline materials

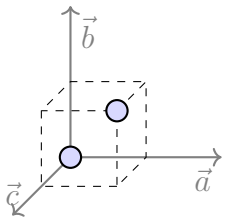
Materials in solid state exist in different phases. The phase depends on the arrangement of the constituent molecules. In crystalline phases there is a long-range order in the structure and in amorphous materials there is not. This thesis focuses on crystalline materials.

How is a crystalline structure defined? A perfect crystal is infinite and periodic in each dimension, and does not contain any lattice defects. Since the crystal is infinite in each direction, it is not possible for absolute perfect crystals to exist in nature, however, it is a useful theoretical concept for defining a crystal lattice.

Lattice structures are periodic structures consisting of atoms in a regular arrangement. They can be defined with a set of lattice base vectors and an atom basis. The set of base vectors might be orthogonal and might or might not have the same length across all of the vectors. The lattice vectors are vectors that join the lattice points. The basis itself consists of one or more atoms of the same or different type. The lattice vectors have been visualized in figure 2.1a with a few points that comply with the lattice vectors $\vec{a}$, $\vec{b}$ and $\vec{c}$. Furthermore, the basis has also been visualized in the figure next to that, figure 2.1b, in the case shown in the figure, two identical atoms create the basis together with a separation of $\left(\frac{a}{2}, \frac{b}{2}, \frac{c}{2}\right)$. Before the introduction to different types of lattices, three expressions need to be defined, those are unit cell, primitive cell and conventional unit cell. A unit cell is a cell that has one or more atoms and that is periodically repeated within the lattice. Primitive and conventional unit cells are both unit cells, but they might be different from one another. A primitive cell is the unit cell of the smallest volume possible. It might have shape and lattice vectors different from the conventional unit cell. A conventional unit cell is the smallest cell that preserves all the symmetries of the lattice and, as previously mentioned, may or may not be identical to the primitive cell. As an example of the difference between the primitive cell and the conventional unit cell we can consider body centered cubic lattice structure (bcc), the bcc primitive cell has one atom and non-orthogonal lattice vectors, but the conventional unit cell of bcc has two atoms and orthogonal, same lengthed lattice vectors. This difference between the bcc primitive and conventional unit cells is shown in figure 2.2.

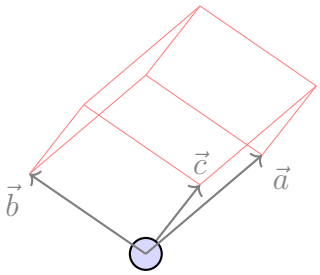


(a) *Lattice vectors and lattice points, the color indicates the coordinate in the direction of* $\vec{b}$

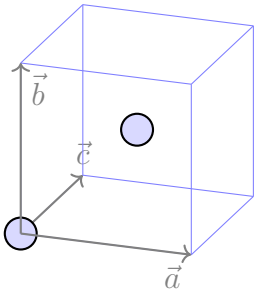


(b) *Basis*

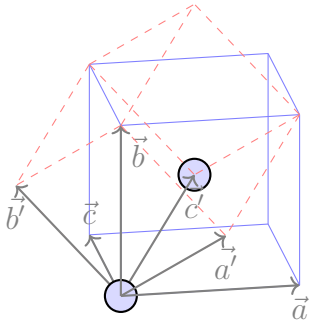
Figure 2.1. *Lattice and basis*



(a) *Primitive cell*

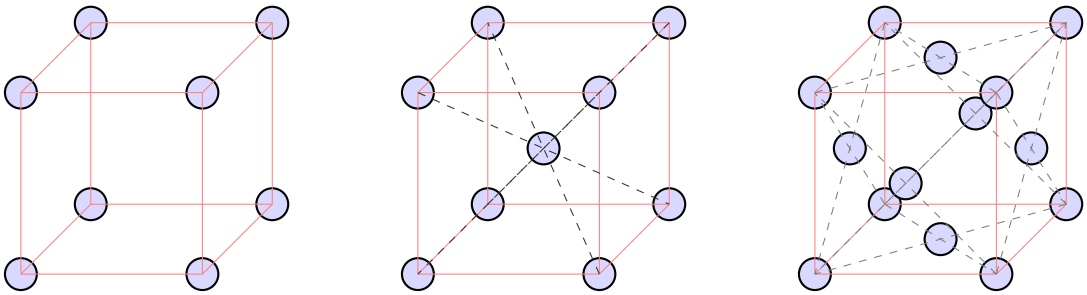


(b) *Conventional unit cell*



(c) *Both in relation to each other*

Figure 2.2. *Primitive and conventional unit cell of bcc*

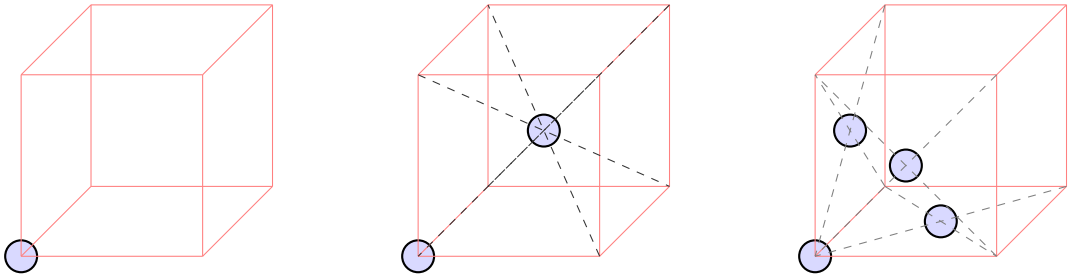


(a) *Simple cubic lattice*

(b) *Body centered cubic lattice*

(c) *Face centered cubic lattice*

Figure 2.3. *Cubic lattice types*



(a) *Simple cubic*

(b) *Body centered cubic*

(c) *Face centered cubic*

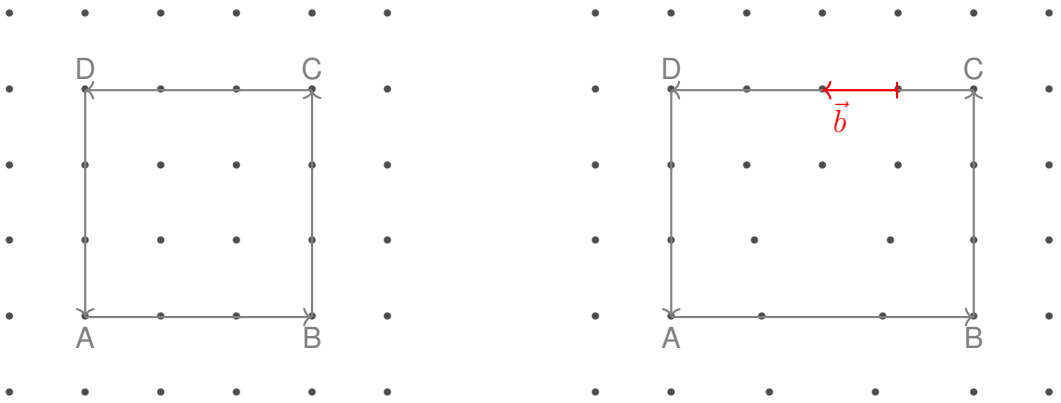
Figure 2.4. *Cubic lattice unit cells*

2.1.1 Cubic lattices

There are three types of cubic lattice structures, which are introduced next. First and the simplest one of them is the simple cubic lattice. Its structure is shown in figure 2.3a. In this thesis conventional unit cells are used to visualise differences between different cubic lattice types. Simple cubic lattice structure (sc) is, as the name suggests, the simplest lattice structure out of the cubic lattices. The conventional unit cell of the simple cubic lattice structure consists of only one atom. The next cubic lattice is body centered cubic lattice structure (bcc), which is illustrated in figure 2.3b. The difference between the conventional unit cell of bcc lattice and sc lattice is the one additional atom in the middle. This makes the bcc lattice type more dense if sc and bcc have the same lattice constant. As seen in figure 2.2a the primitive cell of bcc contains only one atom, but the base vectors are not orthogonal [13].

The third cubic lattice type is the face centered cubic lattice structure (fcc). There is quite a bit of difference between sc and fcc lattices. The conventional unit cell of fcc has 4 atoms, which is more than bcc and sc. fcc lattice can be seen in figure 2.3c.

The unit cells of all three cubic lattices are illustrated in figure 2.4. By repeating the unit cell in all three directions along the edges of the cubic cell shown in red we can create periodic lattices of simple cubic, body centered cubic or face centered cubic lattice types.



(a) *Perfect lattice* ***(b)*** *Edge dislocation*

Figure 2.5. *Defining a Burgers vector of an edge dislocation*

2.2 Lattice defects

In nature crystalline solids are never perfect lattices of identical atoms. The lattice structure may include deviations from the perfect lattice structure, so called defects. Defects might be for example missing or extraneous atoms, misplaced atoms or misplaced portions of lattice. The defects can be divided into different types depending on the number of dimensions they extend in.

2.2.1 Point defects

First we discuss the zero-dimensional defects, so called point defects. The defects in zero dimensions are either missing or extraneous atoms compared to the perfect crystalline structure. They are further classified into different types depending on the location of the missing or extraneous atom.

Lattice vacancies are simply missing atoms in lattice. Depending on where the missing atom resides the lattice vacancy can be defined as Schottky vacancy or Frenkel defect. Schottky vacancies are lattice vacancies, where the missing atom from the lattice can be found on the surface of the material.

If we add extraneous atoms somewhere within the lattice that would not be occupied by an atom in the perfect lattice, we get an interstitial position. If a lattice vacancy atom has been moved in to an interstitial position, we can call the pair of point defects Frenkel defects.

2.2.2 Dislocations (Line defects)

Dislocations, one type of line defects, are lines around which the atoms are misaligned. There are two main types of dislocations: edge and screw dislocations.

The direction and the magnitude of the lattice distortion is defined with a Burgers vector. All dislocations are either edge dislocations, screw dislocations or some kind of combination of these two. For pure screw dislocations, the Burgers vector is parallel to the dislocation line, and for pure edge dislocations it is perpendicular. For a dislocation which is a combination of edge and screw dislocation, the Burgers vector is neither parallel nor perpendicular to the dislocation line. One way for measuring the Burgers vector is to compare a loop in a perfect lattice and around a dislocation line. A loop that in a perfect lattice is closed will be open around a dislocation line. The vector necessary to close the loop in the latter case is called the Burgers vector. Figure 2.5 illustrates the definition of the Burgers vector $\vec{b}$ in an edge dislocation. [13] Stress-driven dislocation movement is one of the mechanisms to induce plastic deformation of crystals. The plastic deformation of a material with dislocations is easier than material with perfect lattice. The plane along which the dislocation moves is called a slip plane.

2.2.3 Plane defects

Plane defects occur along a two-dimensional plane within or in the surface of the material. The plane defects can be divided into three distinct categories: surfaces, stacking faults and grain boundaries. All real crystals are finite, which results in the existence of surfaces which are one plane defect type.

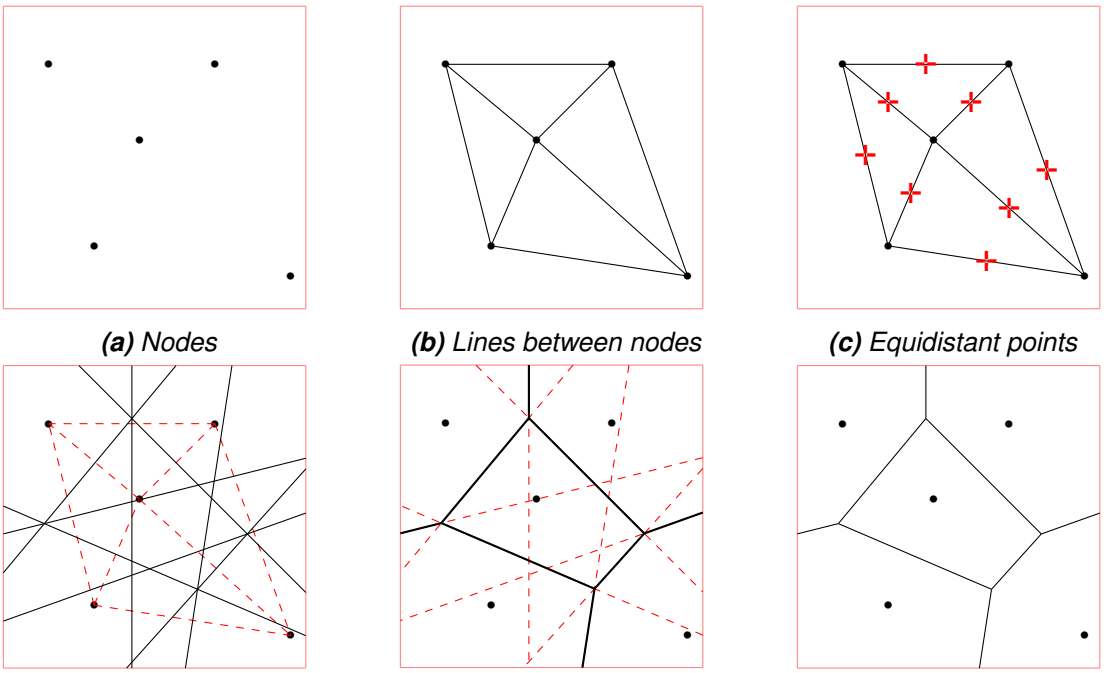
The second type of plane defects, the stacking fault occurs in structures that have more than one possible position for the planes in the structure. For example fcc has a plane stacking sequence $\cdots ABCABC \cdots$, which could be disturbed by one differently placed plane, which would then make the stacking sequence of planes $\cdots ABCABABC \cdots$.

The third type of plane defect considered here is grain boundary. Often crystalline structures consist out of multiple periodic parts called grains which differ from each other by a different lattice orientation. The structure is then called a polycrystal. The grain boundaries affect the, especially mechanical, properties of the crystal by decreasing the structural integrity. It is important to notice that a grain boundary is rarely only a single dislocation.

Also three-dimensional lattice defects, such as cracks, pores, precipitates and voids, could be considered, but since three dimensional defects are separated from the rest of the lattice with plane defects, they are not considered a separate class of defects in this case.

2.3 Voronoi tessellation

In this thesis both polycrystalline and monocrystalline materials are studied. The polycrystalline materials are modeled with Voronoi tessellation and consist of polyhedra called



(a) *Nodes* ***(b)*** *Lines between nodes* ***(c)*** *Equidistant points*

(d) *Perpendicular lines* ***(e)*** *Lines trimmed* ***(f)*** *Finished Voronoi diagram*

Figure 2.6. *Voronoi diagram creation*

Voronoi cells as the grains.

Voronoi tessellation is a method of partitioning the space so that every point within the boundary conditions of the space is in a sector with its nearest node. Voronoi tessellation uses nodes as the seed. The nearest node is then searched for all the points in the, to be tessellated, space. By dividing the space this way, in three dimensions the space is divided into polyhedra. The planes separating the polyhedra in 3D would be lines separating polygons in 2D instead.

For easier visualization of Voronoi tessellation we use 2 dimensions in figure 2.6. First we set the positions of the nodes for the Voronoi tessellation (figure 2.6a). In the next two figures 2.6b and 2.6c lines between the Voronoi nodes are created, and the equidistant points for each of the lines are marked with a ✛. Through each of those equidistant points a line is drawn which divides the space into two regions by the closest nodes (figure 2.6d). We can finish the determination of nearest node for each point on the plane by trimming the lines as in figure 2.6e. We then end up with a finished Voronoi diagram as in figure 2.6f.

2.4 Nano/microscale pillar compression experiments and simulations

In pillar compression experiments and simulations a small, usually nano or microscale pillar made out of some material is compressed and the stress within the pillar is mea-

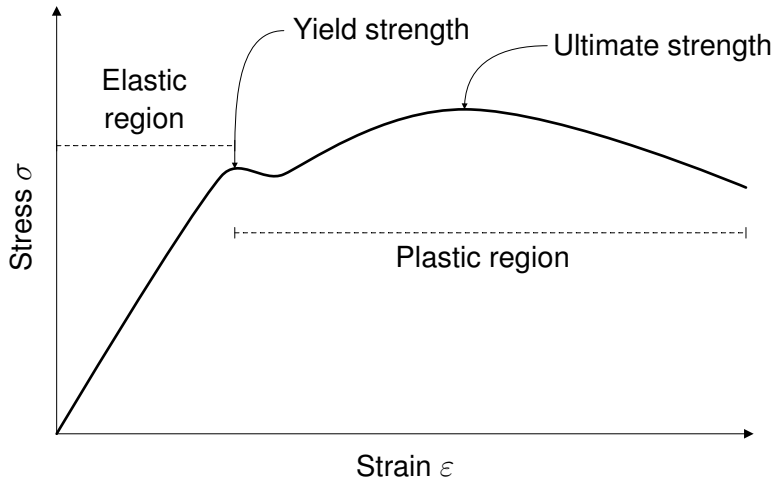


Figure 2.7. *General stress-strain curve*

sured. The experiments and simulations bring information of the different transformation regions, elastic and plastic parts of the stress-strain curve, which is illustrated in figure 2.7. Usually the aforementioned stress-strain curve is the most important outcome of the experiments, but when it comes to simulations, much more data can be acquired easily from the pillar. For example it is possible to save the position of each atom for each single simulation timestep. Even though this is unnecessary in the scope of this thesis' study and would consume a large amount of disk space, it would be possible to trace the exact trajectory of each atom during the simulation. Experiments and simulations differ from one another, a couple of the differences and reasons behind those differences are introduced next.

2.4.1 Experiments

Experiments use precise machinery to carve the pillar out of the base plate of material. The used method might vary from experiment to experiment, but for example focused ion beam (FIB) is sometimes used [14]. The base plate could be for example a monocrystalline piece of material, such as nickel. It might also be polycrystalline sample with grains of varying sizes, but the average grain size can be estimated. [15, 16] Compared to simulations the pillars in experiments have a certain minimum size, which considering this thesis is always greater than the simulated pillar sizes. The minimum possible size of the pillar to machine is a result of the pillar surface contamination during the carving of a pillar with a focused ion beam. Since the surface contamination reaches a certain depth regardless of the pillar diameter, the contamination percentage of the pillar is higher when smaller pillars are carved. [14]

The time scale of the simulation is smaller and the velocity of the indenting planar face is higher than in experiments. For example pillars used in experiments might be approx-

imately 100 times the size of the simulated ones. Experiments might have strain rates in the range of 0.0001 s^{-1} - 0.01 s^{-1}, which is considerably slower than the strain rates of simulations which in case of this thesis are in the range of $2.5 \cdot 10^8$ s^{-1} - $1 \cdot 10^9$ s^{-1}. These difference can not be ignored since strain rates may differ by a factor of 10^{13}. This results in a possibility for large differences in measured physical quantities in direct comparisons between experiments and simulations. There are also pillar compression experiments which do not have a constant strain rate, instead the pillar is strained up to a certain point, after which the indenter stops for a relatively long time, usually for tens of seconds, after which the straining is continued until it is stopped again.[15]

2.4.2 Simulations

Computer simulations are commonly used to study complex systems. There are many different ways to simulate complex systems, such as nanopillars studied in this thesis, for example molecular dynamics (MD), discrete dislocation dynamics (DDD) and Monte Carlo (MC) simulations. DDD simulations, as the name suggests, focus on the dislocations within the material. Kinetic Monte Carlo (kMC) simulations instead use random sampling as is usual for Monte Carlo methods in general. [17, 18, 19] This thesis uses molecular dynamics simulations.

The principles behind molecular dynamics simulations are simple and base on solving numerically the equations of motion for a given system. In each timestep the force acting on each particle is determined and subsequently the positions, and the velocities of all the particles in the system are updated by an integration algorithm. The same procedure is repeated for each timestep of the simulation. As an example of molecular dynamics, we can define a system with N atoms, which collide with each other. For each simulation (time)step, the gravity between the particles, the electromagnetic forces, possible boundary collision forces, and the interatomic forces are summed, the forces then together with the mass of the particle affect its velocity, and the velocity together with the timestep affects the position of the particle. This method of simulation is done numerically together with some time integration method.

If a program calculated all the forces between all the atoms, the calculation time of one timestep would belong in $\mathcal{O}(n^2)$ complexity. If the number of simulated atoms is doubled, the time needed to calculate the timestep would quadruple. This is why molecular dynamics simulations use a maximum range for interaction called cutoff distance. This reduces the complexity of calculation in one timestep to $\mathcal{O}(n)$. With the cutoff distance, linked cell method can be used. In linked cell method the simulation space is divided into cubes with side length being the cutoff distance. This allows for only checking the atoms within the cell and its neighbouring cells to calculate the forces with cutoff distance with. When the cutoff distance with is combined with the linked cell method, the practical efficiency and

speed of the simulation increase.

If computers were infinitely fast and infinitely accurate, anything could be possible to be simulated with accurate results. Since the accuracy of the simulations cannot be guaranteed, they have some additional limitations that experiments do not have. This being said, they also have some advantages over experiments. Due to computational resource limitations, it is not practical to use strain rates lower than certain rate. Here computational resources mean both time and spatial requirements, for example CPU number and time, RAM amount and disk space requirements. This strain rate is still orders of magnitude higher when compared to experiments, especially if the pillars simulated have the same size. Adding to the same point with practical maximum for computational resource usage, it is not possible to simulate pillars that have size similar to those in experiments. Even thought it is possible to slice a simulation to multiple smaller parts of the same simulation, there is a practical limit on how long it is possible to run a single simulation. In some occasions that might be a result of the file system used, or a result of the possibility of a file corruption becoming too great. There is also the fact that a simulation can use more cores to perform the simulation more quickly by utilizing parallel computing, but for a simulation of a specific sized pillar there is a point of diminishing returns in wall clock time for increasing core number for the simulation. Different simulation types deal with different size and time scales. Molecular dynamics simulations are in the scale of single atoms. The simulations can be performed on nanopillars with different sizes, and are reproducible unlike experiments, which always have some fluctuation due to the measurement being affected by the real world outside the measurement equipment. The simulations can also gather a lot more data from a single simulation. For example it is possible to save the position of each atom on each simulation step.

2.4.3 Creating the nanopillar

When creating a polycrystalline nanopillar with a specific average grain size, we first need to calculate the approximate number of grains required to obtain such average grain size. First we need to derive an equation for the grain number, and we start it with an average grain size and number of grains N.

$$N \cdot V_{\text{average}} = V_{\text{total}} \tag{2.1}$$

Since the grain size is a one-dimensional quantity, it is defined in this thesis as an average diameter d of a spherical grain. With that definition, the equation (2.1) is reformulated to

$$N \cdot \frac{4 \cdot \pi \cdot r^3}{3} = V_{\text{total}}, \tag{2.2}$$

where the radius r is the radius of the spherical grain. And from here we change the radius to the diameter

$$N \cdot \frac{4 \cdot \pi \cdot d^3}{3 \cdot 2^3} = V_{\text{total}}. \tag{2.3}$$

After solving the equation for N, we find that

$$N = \frac{6 \cdot V_{\text{total}}}{\pi \cdot d^3}. \tag{2.4}$$

The equation can then be used to calculate the necessary grain number for a grain size if the volume of the generated material is known.

3. NANOCRYSTAL GENERATOR

3.1 Nanocrystal generator

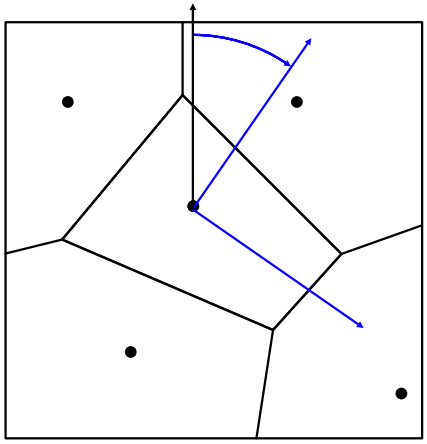
For the purposes of this thesis a new program for generating monocrystalline and polycrystalline materials was written. With that program it is possible to create both bulk material and nanopillars which is the focus of this thesis. The program, currently called Nanocrystal generator [20], has been written by the author of this thesis, Eerik Voimanen. C++ has been used to write this program, and it uses a few different libraries to for example calculate Voronoi tessellation vertex points, to perform linear algebra operations, and to parse command line arguments. These libraries in question are CLI11 [21] for command line parsing, Eigen [22] for linear algebra and Voro++ [23] for some of the Voronoi tessellation related calculations. The program has some advantages over Atomsk, which is "The Swiss-army knife of atomic simulations" – a program that is able to generate initial configurations in addition to other things when considering the scope of this thesis [12]. Those advantages are introduced in this chapter.

3.2 Generation of polycrystalline bulk material

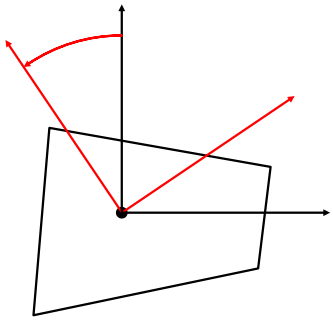
Structure of metallic materials was already introduced in section 2.1, and furthermore the concept of Voronoi tessellation was introduced in section 2.3. Voronoi tessellation is implemented together with the concept of basis and lattice vectors and intrinsic Euler angles to create randomly oriented lattices for different grains in the polycrystalline material.

The properties of Voronoi cells, specifically the vertex points between the Voronoi cells are utilized to calculate the minimum possible size for a cuboid, which can be then tried to be filled with atoms, and for each of the atoms, the atom position is also checked to be within the boundaries of the correct Voronoi cell.

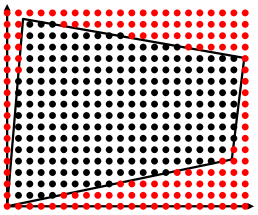
The following method, which is used in Nanocrystal generator to generate rotated lattices within Voronoi cells in 3D is visualized as 2D in figure 3.1. The cuboid is calculated by first rotating the whole Voronoi cell by an inverse rotation matrix of the final rotation of the lattice. Then the maximum and minimum coordinates for each X, Y and Z coordinates are subtracted from each other, and that gives us the side lengths of the minimum cuboid. After the cuboid calculation, the atoms are created, and for each of the atoms, the non-



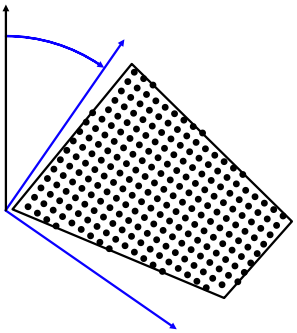
(a) *Target lattice orientation in the middle Voronoi cell*



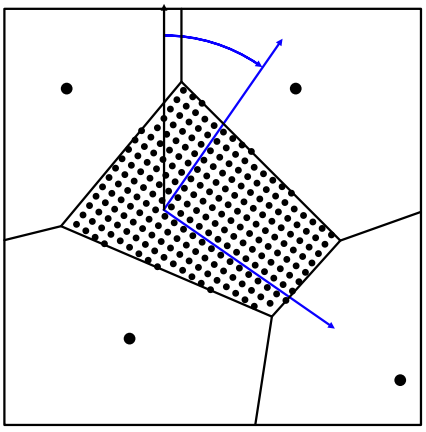
(b) *Rotate the cell with an inverse rotation*



(c) *Generate the atoms with non rotated lattice*



(d) *Rotate the generated atoms and only save the necessary ones*



(e) *Rotated lattice within the cell in original tessellated space*

Figure 3.1. *Creating a rotated lattice within a Voronoi cell*

inverted rotation and translation to the correct location is done. When generating pillars, there is also an additional check, which is introduced in 3.3. After that the correctness of the Voronoi cell for the atom is tested. If the atom is being determined to be within the volume of the Voronoi cell currently being created, the atom is then added to a data structure, which includes atoms. After the position of the atoms inside all of the Voronoi cells have been calculated, the resulting data structure is then output to the LAMMPS data file.

3.3 Optimization for creating pillar shapes

Since this thesis studies nanopillars, the Nanocrystal generator was additionally optimized for the creation of nanopillars in addition to bulk material. The optimization for creating a certain shaped nanopillars has been achieved by implementing a different method than Atomsk utilizes.

For Atomsk a generation of nanopillar is done by first creating the whole bulk material, which is then cut to a pillar by selecting and removing atoms. Since Atomsk does not have a feature especially designed for generating pillars, this is the only way pillars could be created. This requires unnecessary computations, since for creating multiple nanopillars a more efficient way can be utilized.

Nanocrystal generator checks if each atom is within the nanopillar for each of the Voronoi cells, and since the program only saves positions of the atoms that have passed the checks, the required RAM amount for creating a nanopillar from a large bulk material is decreased compared to Atomsk since the whole bulk material does not need to be stored in the RAM.

3.4 Input files

It is possible to input settings, such as atom type, lattice type and boundaries, for Nanocrystal generator with command line arguments or with a partially Atomsk compatible `.params.txt` file. The file includes a description of the box boundaries with each node with the respective Euler angles for the node. Nodes in this case represent the Voronoi points, which are used to create the tessellation of the polycrystalline material. Since the file includes the box boundaries the program is able to get the boundary box dimensions from the file. If a nanopillar is to be made, the dimensions of the pillar need to be inputted in addition to the `.params.txt` file, since the file does not include that information. The file all in all includes the bulk material piece dimensions and Voronoi points and Euler angles of the lattice for each of the Voronoi points. The Euler angles in Nanocrystal generator are currently specified to perform rotations around X, Y and Z axis in that order.

3.5 Output files

At the time of writing of this thesis, the program outputs the whole generated crystalline structure only in LAMMPS compatible format [24]. The output file name is always required to be given fully with an optional filename extension `.lmp` to the parameter `--filename` or `-f`.

In addition to the generated crystalline structure, the program outputs two other files. One of those files is the `.params.txt` file already described in 3.4.

The second additional file which the program outputs is `.Voronoi` file. This file is in LAMMPS format, which includes all the Voronoi points as atoms, for easy visualisation of Voronoi points for example with Ovito. The masses of those Voronoi point atoms are the same as the mass of the generated atom type.

3.6 Advantages over Atomsk

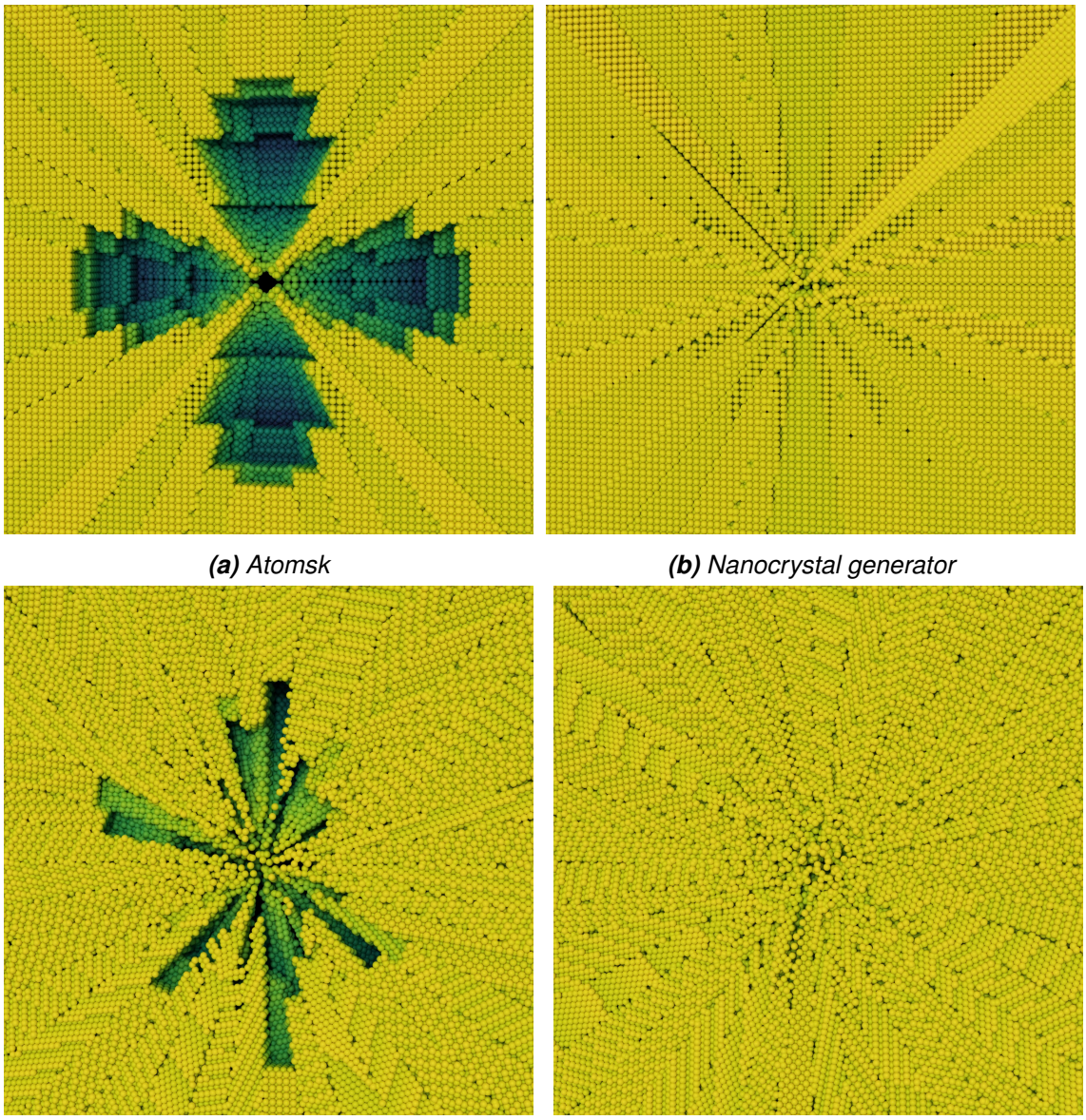
3.6.1 Functionality

When comparing the functionality of this polycrystalline bulk material generation with Atomsk, it is notable that there is no possibility for voids to be generated within the bulk materials.

Atomsk approximates Voronoi cells with template grains. Template grains have approximated dimensions, which in some cases have been not large enough, resulting in voids within the polycrystalline structure. Even thought the templating algorithm will often probably work and could in some cases be faster to calculate than the Voronoi cell vertex point method of the Nanocrystal generator program, the method Nanocrystal generator uses will always generate enough atoms to not leave any voids within or between grains. At the time of writing this thesis, the latest release of Atomsk is b0.11.2, after which there exists at least one GitHub commit mentioning possibilities of voids being generated due to the insufficient template size.

While doing research, an additional bug was found in an older version of Atomsk. There was a possibility, for lattices from multiple Voronoi grains to coexist in the same spatial locations. The reason was the incorrect trimming of Voronoi grains. In the meantime the issue has been reported and fixed for 2D case, which fixed the bug also in 3D.

Atomsk also has problems interpreting the Euler angles from the polycrystal mode node file correctly. The Euler angles of Nanocrystal generator have been confirmed to match with the output of Ovito [2], after considering lattice rotation symmetry. If the Euler angles from Ovito are used to recreate the piece of material, the output of Nanocrystal generator will have the same Euler angles for the lattices inside different grains.



(a) *Atomsk*

(b) *Nanocrystal generator*

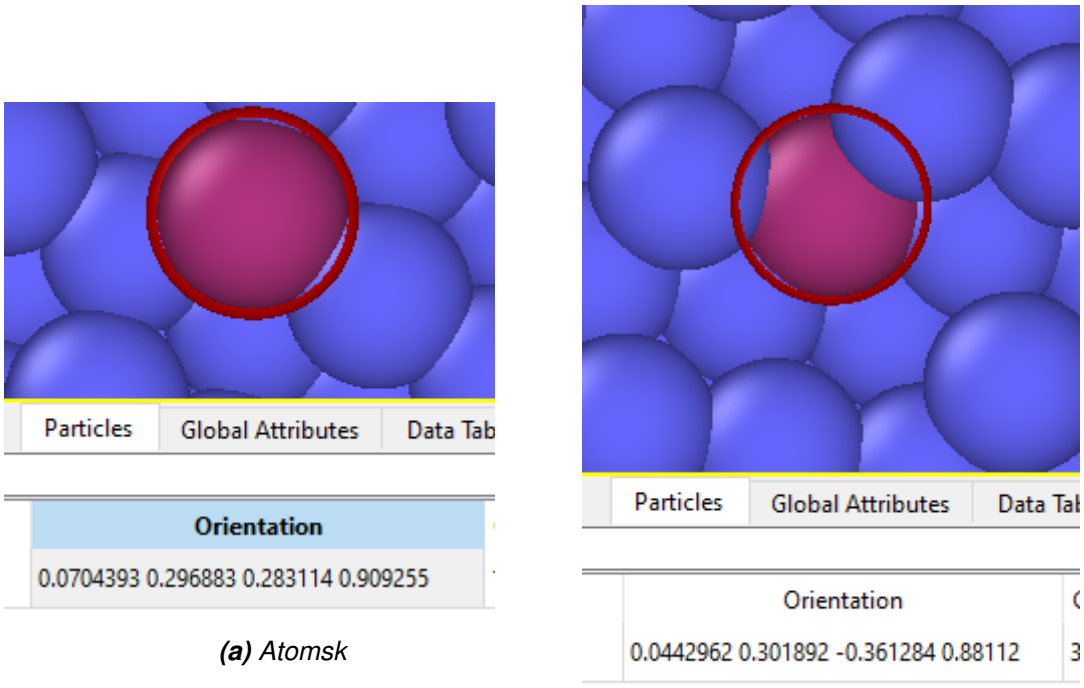
(c) *Atomsk, random lattice orientation*

(d) *Nanocrystal generator, random lattice orientation*

Figure 3.2. *Specific configuration to generate voids in Atomsk*

The previously mentioned partial compatibility with Atomsk `.params.txt` files comes from the fact that even thought the documentation of Atomsk states the Euler angles to use the same format as Nanocrystal generator, the output has never been corresponding to the Euler angle definitions of the documentation. If the Euler angles of the Atomsk polycrystal lattices corresponded to the documentation order XYZ, the `.params.txt` file of Nanocrystal generator should be compatible to be used with `--polycrystal`-mode polycrystal definition file of Atomsk and compatible the other way around if only keywords `box` and `node` are used with degrees without the degree symbol (degree symbols are optional in Atomsk, not allowed in Nanocrystal generator).

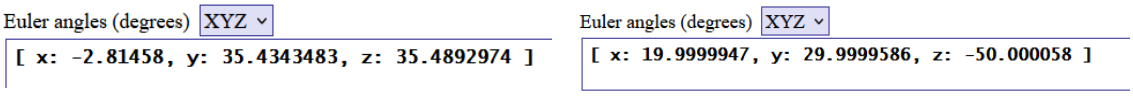
As a demonstration of this inaccurate behaviour, let's create monocrystals with the follow-



(a) *Atomsk*

(b) *Nanocrystal generator*

Figure 3.3. *Quaternions from output of Atomsk and Nanocrystal generator created with input angles* $(X, Y, Z)(20°, 30°, 40°)$



(a) *Atomsk*

(b) *Nanocrystal generator*

Figure 3.4. *Euler angles of Atomsk and Nanocrystal generator created with input angles* $(X, Y, Z)(20°, 30°, 40°)$

ing data as the parameters for the nodes:

```
box 100 100 100
node 50 50 50 20 30 40
```

According to the Atomsk documentation, the node has six parameters which in this particular case are coordinates (X, Y, Z) and three angles in degrees. Atomsk documentation defines these angles with: "The seed will be rotated by the first angle around the cartesian X axis, then by the second angle around Y, and finally by the third angle around Z." The same should hold true also for Nanocrystal generator. This results in Euler angles in

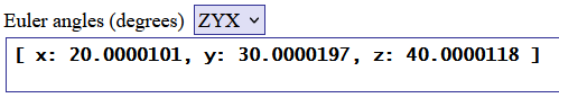


Figure 3.5. *Euler angles of Atomsk with ZYX order*

the order X Y Z. After creating these monocrystals, they were opened in Ovito, and then polyhedral template matching [25] modifier was used with other modifiers to examine the Euler angles of the structure in the middle of the monocrystal. This was done to make sure that the orientation of the crystal was measured from the middle of the crystalline material. The edges did not necessarily obey the periodic boundary condition perfectly. This was the case if the orientation of the atoms made the boundary distances indivisible by the lattice vectors in that direction, resulting in invalid lattice structure planes. Ovito returns quaternions, which are completely different between the two programs, as can be seen in figure 3.3. The quaternions can be converted to Euler angles, and for this purpose, an online tool was used [26, 27]. With that online tool by putting in the quaternions extracted from monocrystals generated by both programs, we get output which is shown in figure 3.4. Since bcc has a rotational symmetry of 90° around each crystallographic axis, rotations differing by a multiple of $N \cdot 90^\circ$ are equivalent. We can notice that Nanocrystal generator has rotated the basis correctly. For Atomsk, the angles are incorrect, but if one changes the order of rotations to be ZYX instead, the result is correct as seen in figure 3.5.

Due to possibility of having a void within a polycrystalline structure and the fact that Euler angles do not match with the documentation, this thesis considers Nanocrystal generator being more accurate at creating polycrystalline bulk materials than Atomsk. The possibility of a void existing within a polycrystal created with Atomsk has clearly been reduced with the previously mentioned commit, but since Atomsk uses templates, the possibility of a void might not be zero in some specific circumstances.

3.6.2 Performance

The performance tests for Atomsk and Nanocrystal generator are carried out in order to verify if there is some kind of correlation between the generation time, bulk material dimensions and number of grains. Also the correlation between memory usage, bulk material dimensions and number of grains is observed.

To compare the performance between Nanocrystal generator and Atomsk, a few different tests with different aims have been made. To stabilize as many variables as possible, the performance tests have all been run on CSC's Puhti cluster. Tests have been given 8 cores and 192 GB of RAM to work with. In addition to this, since the way the Euler angles are calculated works differently between the programs, for each combination of parameters, three different random configurations have been used. These should result in more accurate statistical approach to the performance analysis. Due to differences between the programs and the possibility of difference in scalability with fewer or more cores, it is not possible to state the absolute performance with this kind of testing which only uses a single type of processor, single fixed number of cores and one specific maximum amount

of RAM.

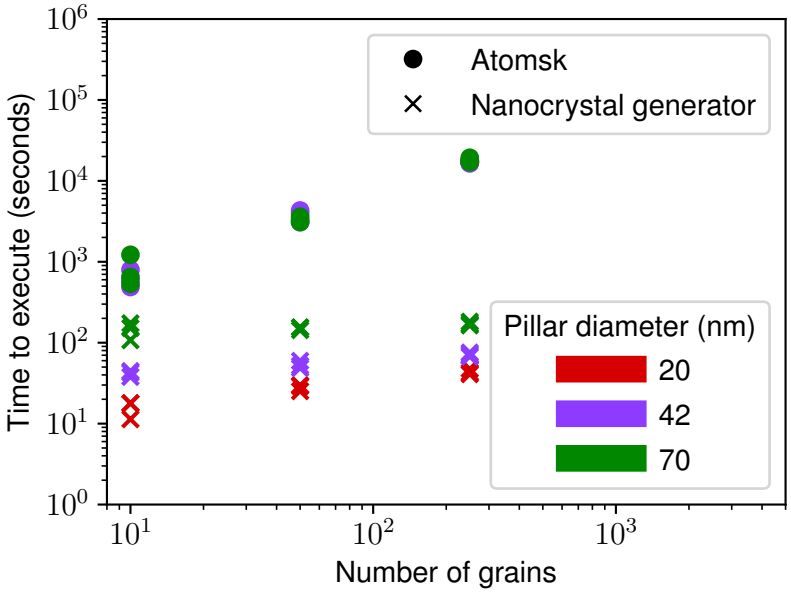
The performance comparison will be started with nanopillars since the research in this thesis considers those and a lot of nanopillars are generated for the simulations. A set of tests between different sized and different grain number nanopillar generations have been made. All combinations of three different midway diameters, 20 nm, 42 nm and 70 nm and three different numbers of grains 10, 50 and 250 are generated. Since even if the pillars are smaller than the maximum pillar size the bulk material can handle, they will be fully generated, only a single bulk material size has been used. The bulk material size has been selected to be 180 nm by 180 nm by 210 nm and to utilize periodic boundary conditions.

Due to the fact that nanopillars are not possible to be generated in Atomsk without also generating the whole bulk material polycrystal, it takes more time for Atomsk than Nanocrystal generator to generate those. This is quite evident in the top plot of figure 3.6. There is a clear scaling of generation time with the number of grains in Atomsk, but the scaling is more modest in Nanocrystal generator. For the scaling of time for pillar diameter, due to the way nanopillars are created, there is no clear scaling for Atomsk but there is a scaling for Nanocrystal generator.

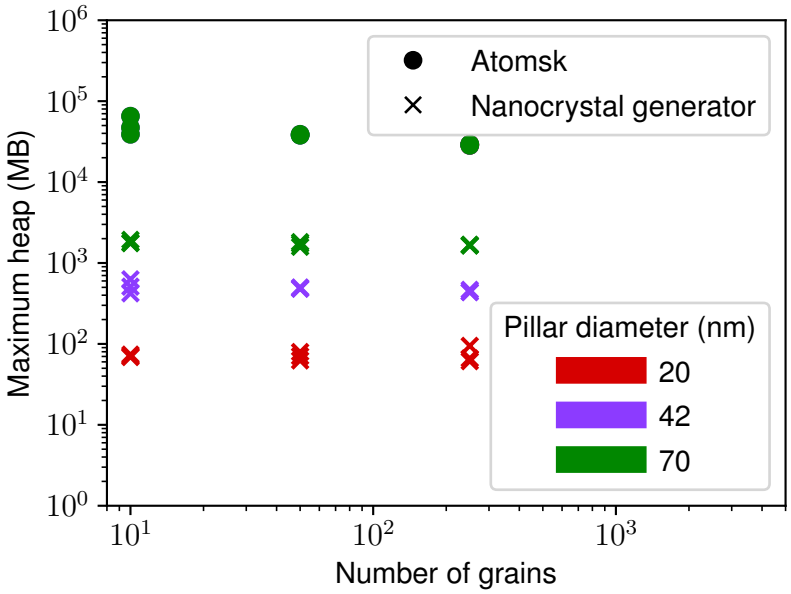
In the bottom plot of the same figure, the values for maximum RAM usages are shown, and it is easy to notice that Nanocrystal generator seems to take up a lot less memory for these specific tests. Also the amount of RAM used by Nanocrystal generator does not scale according to the number of grains, but for Atomsk there is a lowering trend for memory usage for larger numbers of grains. The memory usage of Nanocrystal generator seems to scale for the pillar diameter, but Atomsk seems to not do this. This is due to the same reason already mentioned in time scaling, the nanopillar generation is done with a slower method on Atomsk.

It is not clear from the figures, but there are 6 missing points for Atomsk, this is due to the jobs terminating before the program was finished or the program crashing. No clear pattern between the initial configurations and these failed runs was found. For Nanocrystal generator there were no failed runs.

Due to the difference between the way Atomsk and Nanocrystal generator generate nanopillars a more fair comparison of performance between these two programs is to generate a bulk material of certain size and with a certain number of grains. In that case, the method of generating the grains differs, but both programs need to generate the same bulk material without any major shortcuts. Due to the fact that Atomsk does not understand Euler angles the way it is stated in the documentation, it is not possible to create identical crystalline structures with both programs. For this reason three different random configurations were used to minimize the possibility of favouring one of the programs.



(a) *Generation time*

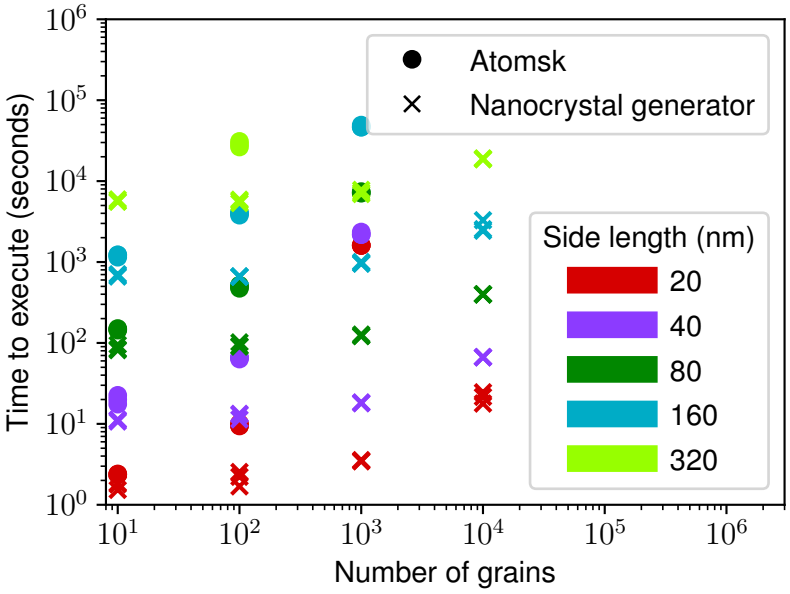


(b) *Maximum heap memory of generation*

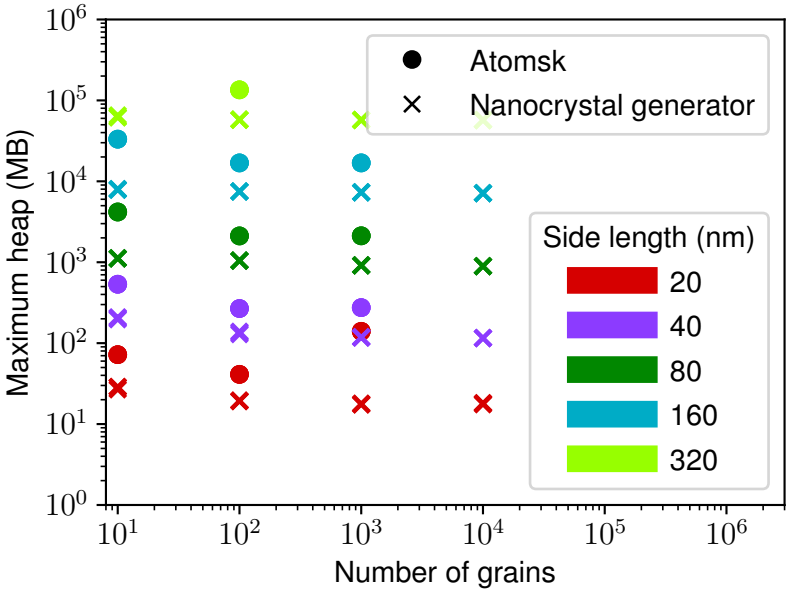
Figure 3.6. *Nanopillar generation efficiencies of Atomsk and Nanocrystal generator*

For the comparison, polycrystalline cubes have been generated with all combinations of the given side lengths and numbers of grains. For the bulk material side lengths 20 nm, 40 nm, 80 nm, 160 nm and 320 nm are used. And for the the number of grains 10, 100 and 1000, 10000 grains are used. With three different random configurations for each combination this results in $5 \cdot 4 \cdot 3 = 60$ different generations for both of the programs.

In figure set 3.7 we can find the data from these types of bulk generation runs. It is possible to note the absolute memory usage being approximately half on Nanocrystal generator when comparing to Atomsk. We can also note that both Atomsk and Nanocrystal generator seem to have no scaling for the memory usage in respect to the number of



(a) *Generation time*



(b) *Maximum heap memory of generation*

Figure 3.7. *Bulk material cube generation efficiencies of Atomsk and Nanocrystal generator*

grains. It is also possible to note that for both programs there is a clear scaling for memory usage in respect for side length of the cube.

For the time it is possible to see an increase with respect to grain number for both programs. For Atomsk the time seems to increase more when compared to Nanocrystal generator. The faster increase of runtime with respect to number of grains is expected for Atomsk, since there is a number of functions that are run for each of the grains and one of them is the shifting of mass center point. For this particular use case the shifting of all atoms to move the center point of the bulk material after each grain has been generated is unnecessary due to it having no effect on the end result. For both programs, the time clearly scales with the side length of the bulk material. In these figures one can clearly see at least some missing entries for Atomsk. Atomsk did not finish any of the 10 000 grain bulk material generations due to time or memory constraints or due to crashing of the program. In total Atomsk had a total of 21 failed runs. A large portion of the runs ended due to the time limit of Slurm jobs, which was set to be 3 days. For these jobs it is not possible to determine whether the jobs had just barely exceeded the 3 day runtime maximum or if the time limit was exceeded by a lot.

3.7 Nanocrystal generator usage

The usage is demonstrated on an example of a command to create a polycrystalline nanopillar with a midpoint diameter of 20 nm from a bulk material with 40 grains sized 800 Å · 800 Å · 1500 Å:

```
nanocrystal_generator --filename out.lmp --lattice_type bcc \
--box 800 800 1500 --atom_type Ta --grain_amount 40 \
--lattice_constant 3.3026 --seed 42 --diameter 20
```

The first keyword is `--filename`, which sets the output file name without its file extension. In this example, the LAMMPS data file, which has the pillar in it is called `out.lmp`.

The second keyword, `--lattice_type` sets the chosen lattice type. At the time of writing this thesis, the program supports bcc and fcc lattice types, which are introduced with more detail in 2.1. Third keyword `--box` has three arguments. These arguments define the bulk material dimensions (X, Y, Z) in Å.

The fourth keyword, `--atom_type` defines as the name suggests the used atom type. The only reason the program needs the atom type to be given with keyword argument is to convert the atom type to an atomic mass for LAMMPS data file. LAMMPS data files themselves in fact do not require the atom type, but instead do require the atomic masses. The program uses atomic masses from the periodic table published by National Institute of Standards and Technology [28].

Number of grains is specified with the keyword `--grain_amount`, in this case 40 randomly generated points are chosen to be the origins of Voronoi tessellated grains. Voronoi tessellation is introduced with more detail in 2.3. Here it is also important to mention that periodic boundaries are used in this example, even thought the keyword `-p` is not put to the command, the default behaviour of Nanocrystal generator is to use periodic boundary conditions. If one does not want to use periodic boundary conditions for the Voronoi tessellation, it is possible to do that with the keyword `-n`, which disables the periodic boundary conditions.

The next two keywords, `--lattice_constant` and `--seed`, set the lattice constant, which is used as a multiplier for setting the absolute coordinates to the atoms and the seed which is used to initialize the random number generation, which in the case of Nanocrystal generator is Mersenne Twister 19937 pseudo number generation [29]. If one wants to use randomized numbers without having to decide a seed, the seed keyword can be left out, or changed to be the number 0. This makes the program use system time as the seed. In the simulations done in this thesis, the seed has been set by the slurm job id, because even though the jobs might start at the same exact time, making the time initialization of random number generation useless, the job id is always different on each of the jobs.

The last keyword `--diameter` sets the Nanocrystal generator to create a nanopillar instead of the bulk material. Without the parameter the output will be the full bulk material, so no pillar is cut out of it. If the parameter is used, the functionality of the program changes and the memory usage decreases, assuming the bulk material is larger than the nanopillar. This is due to the fact that only a portion of the atoms belonging to the pillar in bulk material needs to be saved.

Even though this example gave the keywords in this specific order, the order of keywords does not matter as long as the values given to the keywords are correct. During writing of this thesis, some additional keywords have been added. Within the scope of this thesis, the most important additional keyword, which is not related to nanopillars is `--no_wrap` which disables periodic boundary wrapping of atoms. With periodic boundary conditions this results in full grains with some of the atoms possibly outside the boundary box. Instead the grains are complete and are not sliced. This keyword is used in 3.6.1 to see how the grains look before wrapping. The possibility to not wrap atoms inside the boundaries helps to compare grain formation between different tools more easily.

Nanopillars also have a few additional measures, namely height and taper angle. As has been done in some other research studies, the height was chosen to be three times the halfway diameter of the nanopillar. Additionally the taper angle was chosen to be 2.5°. At the time of writing this thesis, Nanocrystal generator does not allow for customizing height or taper angle, but uses these dimensions, but allows to choose the halfway diameter of

the nanopillar.

4. MOLECULAR DYNAMICS SIMULATIONS

4.1 Generation of nanocrystalline material

Before running a simulation, a file which includes positions of the atoms in the sample of material needs to be provided for the simulation software. In the case of this thesis, which employs LAMMPS for molecular simulations, LAMMPS data files are created with Nanocrystal generator. Nanocrystal generator's working principles as well as the differences to Atomsk, that could also be used to create a file in LAMMPS format for this thesis have been introduced in section 3.1.

4.2 LAMMPS

4.2.1 LAMMPS basics

LAMMPS is a commonly used open-source software for molecular dynamics simulations. [30] It is possible to compile LAMMPS to use OPENMP and MPI or only one of those to scale to multiple cores and CPUs and in case of MPI also for multiple nodes. The LAMMPS version used in this thesis is one of the recent stable releases called "Update 3 for Stable release September 2021." [1]

LAMMPS includes by default a group of commonly used atomic potentials. In addition to those commonly used potentials LAMMPS also provides a possibility to read the potential from a file. In this thesis the potential file features Extended Finnis-Sinclair potential [31] generously shared to me by Dr. Carlos Ruestes from Universidad Nacional de Cuyo.

LAMMPS simulations use input script files that define the whole simulation with all the inputs and outputs. The definition for the whole simulation includes how the simulation runs and for how long. There are also values for the ensemble, timestep and running time. The first lines of the script specify initial configurations, reading of the LAMMPS data file, which contains the initial (here) coordinates of the atoms within the system, and selection and reading of the potential file. In the input script files it is also possible to limit the movement of one or more atoms by selecting those, set periodic boundary conditions and output data, for example temperature or atomic configuration.

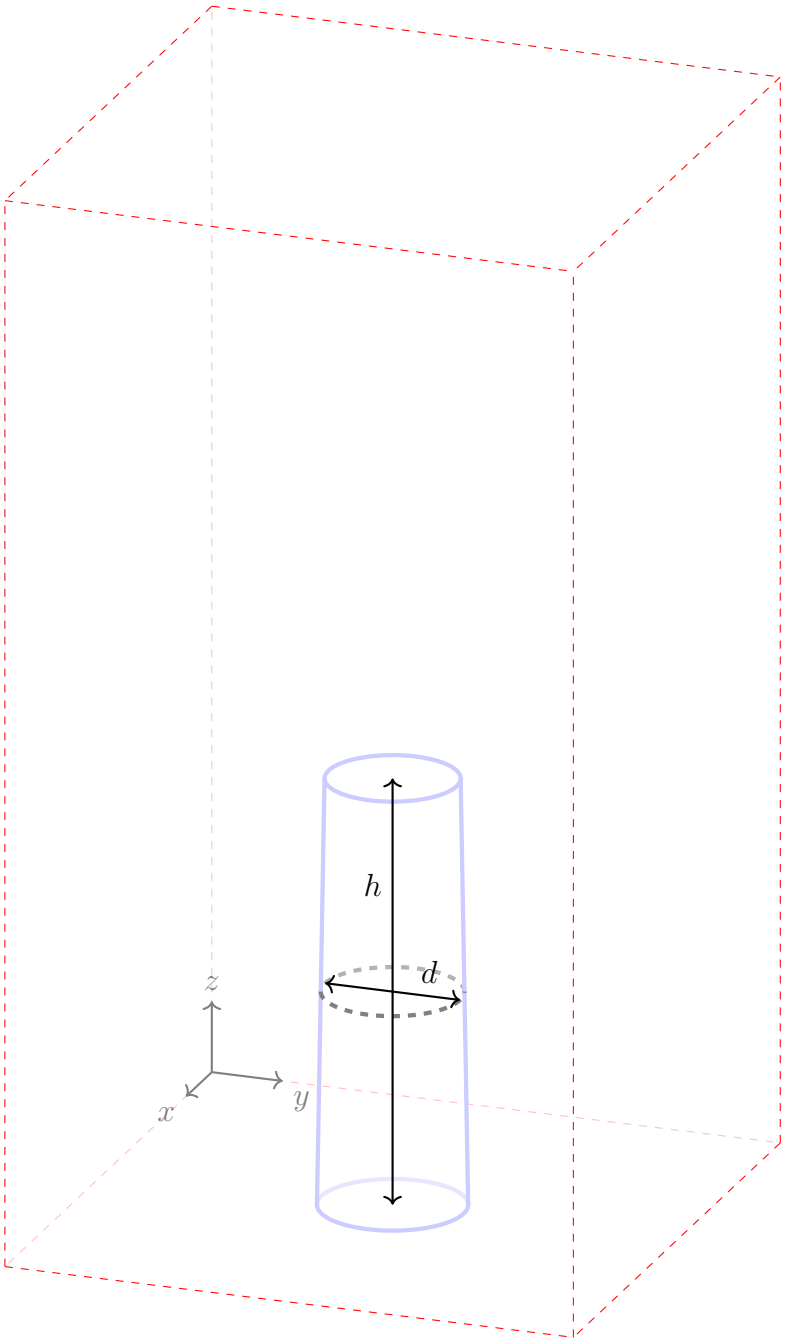


Figure 4.1. *Finalized* 20 nm *nanopillar within the bounding box*

4.2.2 Used features

The length of the timestep of the simulation is defined in the file, in this case it is equal to 1 fs. After reading the initial configuration from the input file, the potential energy of atoms of that pillar is minimized, and then their velocities are initialized randomly at 300 K.

Before the equilibration, the input script defines the indenting plane and fixes the positions of atoms in the two bottom layers, so that the pillar does not move when the indenting plane pushes it towards the negative Z-direction. A visualization of the initial configuration

of the simulation can be found from figure 4.1. The two layers are fixed by first creating a region that includes two layers of atoms and then creating a group of atoms within that region. The script also defines all the atoms outside those two bottom layers as another group, for calculations of the internal stress within the pillar.

Additionally the input script relaxes the pillar for 40 picoseconds in 300 K in an NVT ensemble before the compression stroke which is done in the same temperature. This approach has also been used in an article about micro-mechanical response of tantalum. [8]

During the compression phase the indenting plane is moved at a constant pace of $5\,\mathrm{m\,s^{-1}}$, or $10\,\mathrm{m\,s^{-1}}$, $20\,\mathrm{m\,s^{-1}}$ depending on the simulation. The stress- and thermodynamical data from the simulations is saved with an interval of simulations steps that depends on the height of the pillar. The exact timestep interval Δt is calculated with

$$\Delta t = \frac{v_{\text{indenter}}}{20\,\mathrm{m\,s^{-1}}} \cdot \frac{d_{\text{pillar}}}{20\,\mathrm{nm}} \cdot 5000. \tag{4.1}$$

Using the equation results in approximately the same number of stress-strain curve points for all of the simulations with different indenter velocities and pillar diameters.

Due to some of the simulations being really long, LAMMPS restart feature is also used. For those simulations, a restart file generation interval is defined, and when the simulation timestep reaches a non-zero value divisible by the interval, a binary restart file is saved. The binary file includes the necessary values to restart the simulations, but the other input script which reads the restart file needs to redefine variables such as the ones used to move the indenting plane.

An example of a LAMMPS input script, one script used in this thesis can be found in Appendix A. For example in the first three lines we define units, periodic boundary conditions in X, Y and Z -directions and the style of particles which the simulation will simulate. Since LAMMPS is a molecular dynamics simulation software, which is not restricted to only use single atoms as the particles, there are multiple different particle styles to choose from. [32] In this thesis since the simulations are done with Tantalum, atomic style has been chosen, this means that the particles used in the simulation are atoms.

4.3 Simulation workflow

Since certain features, such as calculating the stress of each atom one by one, in simulations do not scale well to an infinite number of cores, the simulations have been run on either 40 or 80 cores with nodes on CSC's Puhti supercomputer.[33]

With Puhti supercomputer it is possible to select from a number of different batch job running partitions depending on the needs of the simulation. Since the simulation seemed

to scale well to 40 and 80 cores but not beyond that, small and large partitions were used. Since memory usage was not a limiting factor, the "hugemem" partitions were not used. Additionally due to the large number of simulations and smaller number of nodes belonging to longrun partition, it was not possible to run the simulations on longrun partition within reasonable time. [34]

Additionally due to some of the simulations exceeding small and large partition maximum job lengths, it was necessary to find another way of continuing simulations instead of running them on longrun partition. LAMMPS provides this functionality with `restart` and `read_restart` -commands. The command `restart` writes a restart file, which can be read by another simulation job with the command `read_restart`. With the information stored within a LAMMPS binary restart file, the second job is able to continue the simulation as if it was not stopped at any point.

4.4 Creating the compression simulation script

For creating representative statistics it is necessary to collect large amount of simulation data. In order to achieve that it is necessary to run a large number of simulations, which all require input data files and input scripts. In order to improve this workflow by making it less error prone and as efficient as possible a Python script was written. The Python script submits Slurm job arrays which have dependencies with each other. The dependencies within Slurm results in workflow that will start jobs when necessary and if the previous step was completed successfully. With all of this in mind, the aim for the results has been set, and the coding of this automating script was just a matter of time.

In its finished form, with the script it is possible to use only one user input command to generate the Slurm jobs for both generating the initial configuration with Nanocrystal generator and running the simulation itself, possibly in smaller segments. At some point there was also an interest in creating a few different animations of the compression of the pillar and an additional feature was added to also output the configuration in the middle of the simulation and analyze the output after the with Crystal Analysis Tool [35, 36]. With this analyzed configuration together with the stress-strain data directly output from the simulation it was possible to output video files. This feature of additional analysis and video generation was not used with the data in the results section of this thesis due to it being unnecessary for the scope of the research here.

5. RESULTS AND ANALYSIS

5.1 Analysis software

5.1.1 OVITO

OVITO is a widely used atomic configuration visualisation tool. [2] It allows the user to modify the atomic configuration by deleting the visual elements of some of the atoms depending on some kind of variable, for example atom type or coordinates of the atom. It is also possible to perform crystallographic analysis within OVITO.

For this thesis OVITO is used for all of the atomic configuration visualisations. OVITO has useful tools for selecting different atoms by a type, property and a specific variable defined in the input file. It also has good tools for coloring the atoms by those same parameters, so that for example atoms that have coordinate larger than some value can be deleted and the rest can be colored by the coordinate value in some direction.

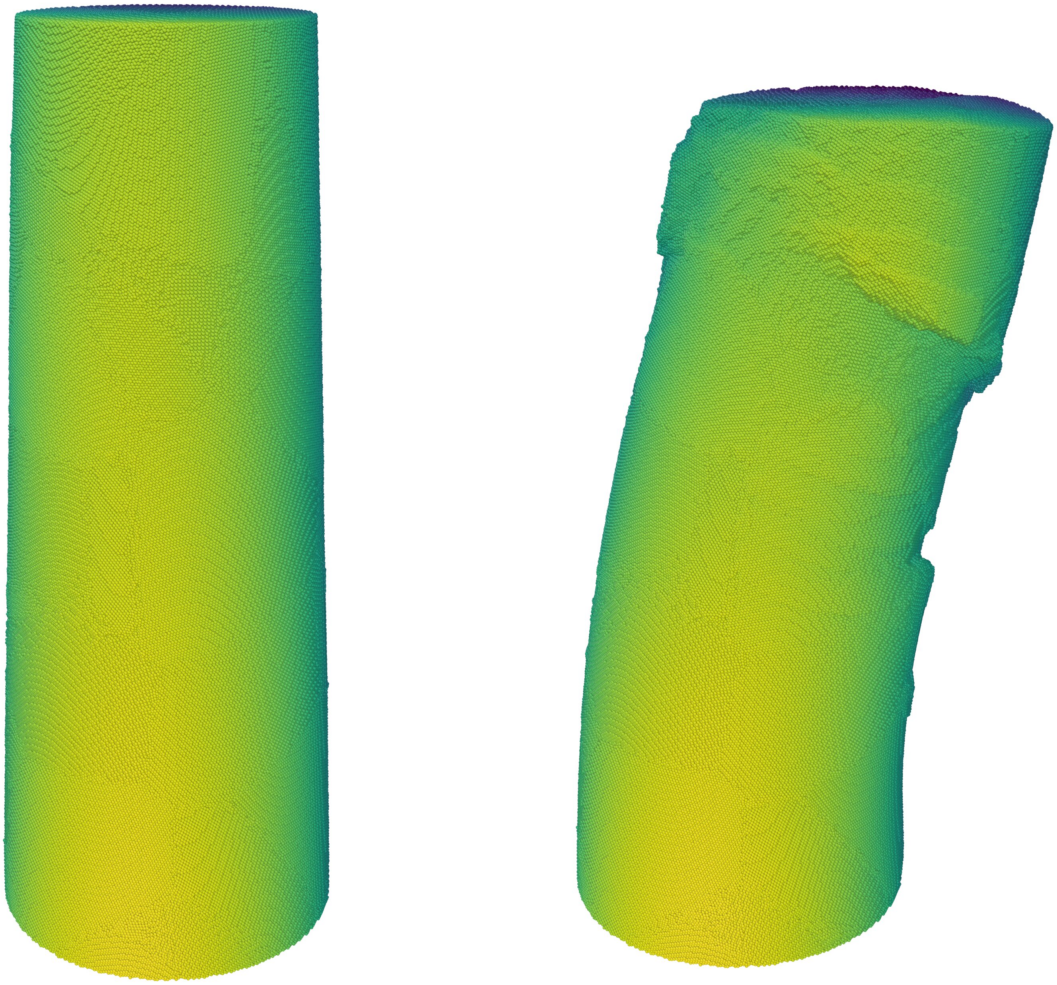
An example workflow of an OVITO visualisation could go as follows:

1. Load a file in OVITO
2. Add expression select modifier, and select portion of the pillar
3. Add second delete selected modifier, so that now it is possible to see inside the pillar
4. Add color modifier so that it is easier to see the topology of the pillar
5. Export the viewport in some figure format

As an example of the output of that workflow, in figure 5.1 there are 3D-renders of the pillars before the compression and after compression. After the compression, the pillar is clearly shorter and deformed.

5.1.2 Python libraries

There are several Python libraries and smaller self-made code snippets that are utilized in this thesis. Starting with libraries, Matplotlib [37] and Seaborn [38], which allow visualisation of data with different types of figures. Additionally NumPy [39], and Pandas [40, 41]



(a) *Before compression* ***(b)*** *After compression*

Figure 5.1. 30 nm *midway diameter cone-shaped tantalum nanopillar with* 20 nm *average grain size before and after 0.12 strain compression in simulation with* 20 m s^{-1} *indenter velocity*

are utilized since those modules have the ability to read and write different file types, calculate statistical variables and are accelerated with natively running machine code which makes the calculation faster.

In chapter 3 OVITO visualisations were used to show the structure of the polycrystalline cube. This was done by utilising the OVITO Python module which also includes some of the features from the OVITO Pro. One example of these features used in this thesis is the ability to generate visualisations of atomic configurations with ray-tracing.

5.2 Tantalum nanopillar statistics

5.2.1 Generation and compression of tantalum nanopillars

Before the simulations a set of LAMMPS data files including initial polycrystalline and monocrystalline tantalum nanopillars was created with Nanocrystal generator. The nanopillars are cone-shaped, with 20 nm midway diameter pillars, a taper angle of 2.5° and a height of 60 nm. A total of 10 different grain sizes in addition to monocrystalline pillars combined with 3 different indenter plane velocities have been used. The used grain sizes were 10, 15, 20, 25, 30, 35, 40, 45, 50, 100 nm. For every grain size, 40 different specimens of pillars were created. For the monocrystalline pillars, the orientation of the lattice was random, and for the polycrystalline pillars the position of the nodes in the Voronoi tessellation were random in addition to the lattice orientation of each grain.

To get a statistical data from 20 nm diameter tantalum nanopillars a large number of simulations has been run. All of the generated pillars were run with three different strain rates, $5\,\mathrm{m\,s^{-1}}$, $10\,\mathrm{m\,s^{-1}}$ and $20\,\mathrm{m\,s^{-1}}$. Since the number of different grain sizes is 11, number of pillars for each grain size is 40 and number of different strain rates is 3, this totals in 1320 simulations worth of stress-strain data from 440 different pillars.

In addition to the 20 nm midway diameter pillars, an additional separate set of pillars has been created. The second set consists of 4 different midway diameter cone-shaped pillars which have a height three times the midway diameter and a taper angle of 2.5°. The selected midway diameters are 20 nm, 30 nm, 40 nm and 50 nm. For all different midway pillar diameters 4 different grain sizes were created. The selected grain sizes were 20 nm, 30 nm, 40 nm and 50 nm. For all combinations of grain size and midway diameter, a set of 12 different pillar specimen were created.

The statistical data of these 20 nm to 50 nm diameter pillars were all simulated with one indenting plane velocity, $20\,\mathrm{m\,s^{-1}}$. The total number of the second set of simulations is 192, which is the same as the number of different nanopillars. These simulations aim to replicate the results in similar to another study on Tantalum nanopillars[8].

5.2.2 Calculation of yield stress

This thesis focuses on the yield stress statistics. Before discussing the statistics, the yield stress needs to be defined. The yield stress is the value of the stress where the deformation behaviour transitions from elastic to plastic. There is no single definition for the yield stress, but in this work the yield stress is calculated with the offset yield stress method with an offset of 1%. The figure 5.2 includes a stress-strain curve for illustrating the method. First a line is drawn by fitting a linear function to the elastic part of the stress-strain curve, here between strains 0.002 and 0.018 and shifting it to the right by 1.0%.

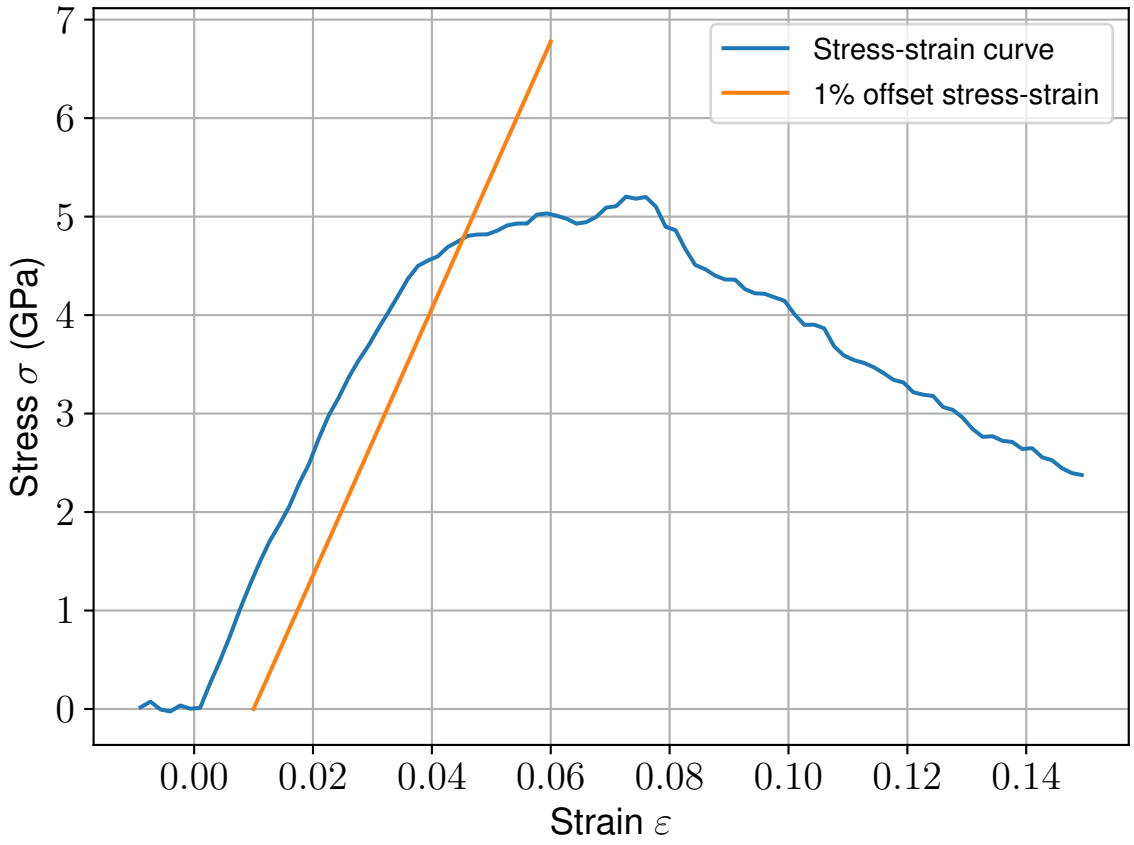


Figure 5.2. *Stress-strain curve and 1.0% offset stress-strain*

The yield stress is then extracted from the intersection point of the stress-strain curve and the offset line. Because the stress-strain curve obtained from the simulation data consists in fact of discrete points, in practice the last point of the stress-strain curve right before the intersection is used. The other way of defining yield stress is using the global maximum of the stress along the stress-strain curve.

5.2.3 20 nm diameter nanopillar statistics

The simulations use straining indenter plane velocity instead of strain rate, but since all the pillars considered in this section have a midway diameter of 20 nm and a height of 60 nm, the selected velocities (20, 10 and 5 m s^{-1}) can be converted to strain rates with

$$\varepsilon(t) = \frac{v(t)}{L_0}, \tag{5.1}$$

where $v(t)$ is the velocity of the indenter plane as a function of time t, L_0 is the height of pillar and $\varepsilon(t)$ is the strain rate as a function of time. For these values the velocity of the indenter plane is constant, resulting in

$$\varepsilon = \frac{v}{L_0}. \tag{5.2}$$

Table 5.1. *Strain rates from indenting plane velocities*

Indenter velocity ($m\,s^{-1}$)	Strain rate (s^{-1})
5	$8.33 \cdot 10^{7}$
10	$1.67 \cdot 10^{8}$
20	$3.33 \cdot 10^{8}$

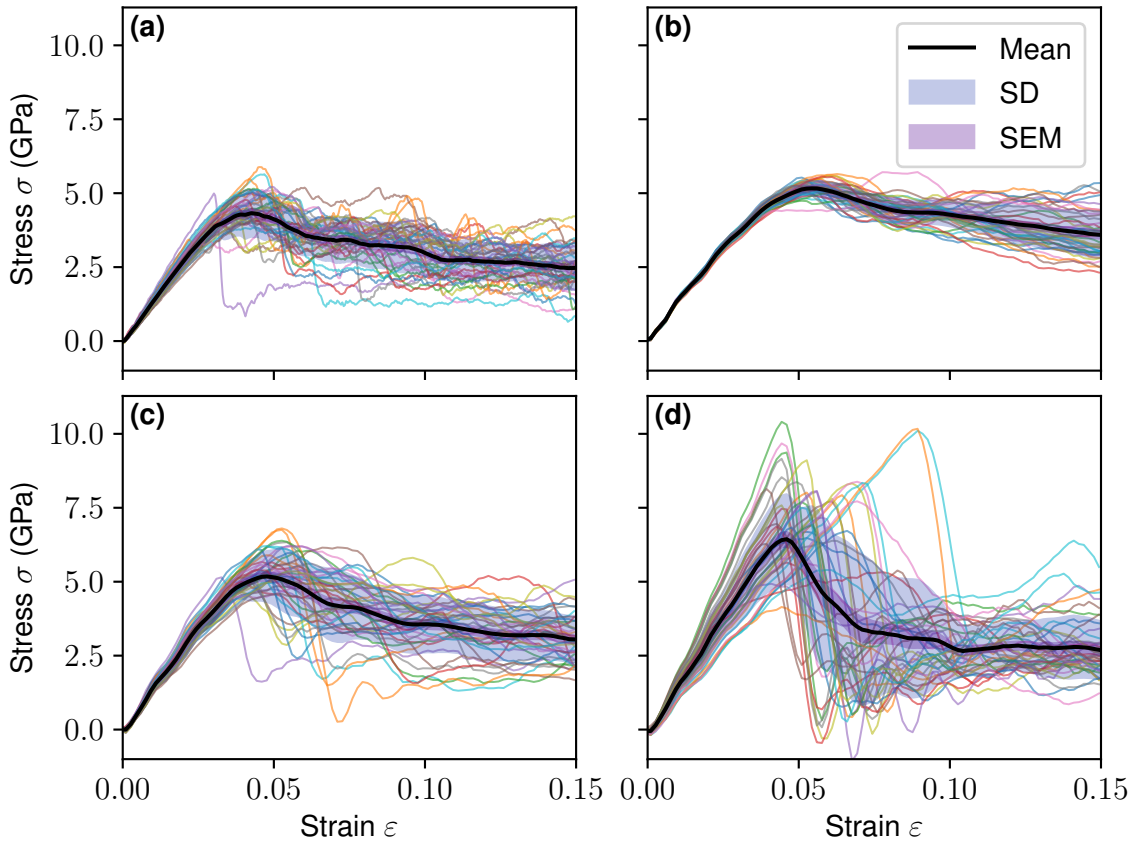


Figure 5.3. *Simulated stress strain curves for* 20 nm *diameter cone-shaped tantalum nanopillars with* ***(a)*** $5\,m\,s^{-1}$ *indenter plane velocity and* 35 nm *average grain size,* ***(b)*** $20\,m\,s^{-1}$ *indenter plane velocity and* 10 nm *average grain size,* ***(c)*** $20\,m\,s^{-1}$ *indenter plane velocity and* 35 nm *average grain size and* ***(d)*** $20\,m\,s^{-1}$ *indenter plane velocity and* 100 nm *average grain size*

In table 5.1 are the corresponding strain rates for different indenter velocities for the nanopillars considered in this section.

First we can look at a few examples of the stress-strain data output by the simulations, visualized in figure 5.3. We can clearly see that the greater the grain size (**(b)** and **(c)** comparison), the greater the difference between individual samples. We can also see from **(a)** and **(c)** that the general roughness of an individual stress-strain curve changes depending on the indenter plane velocity.

We can now have a look at the yield stresses as a function of the grain size. In figure 5.4 there are data with errorbars. The points represent the yield stress values obtained

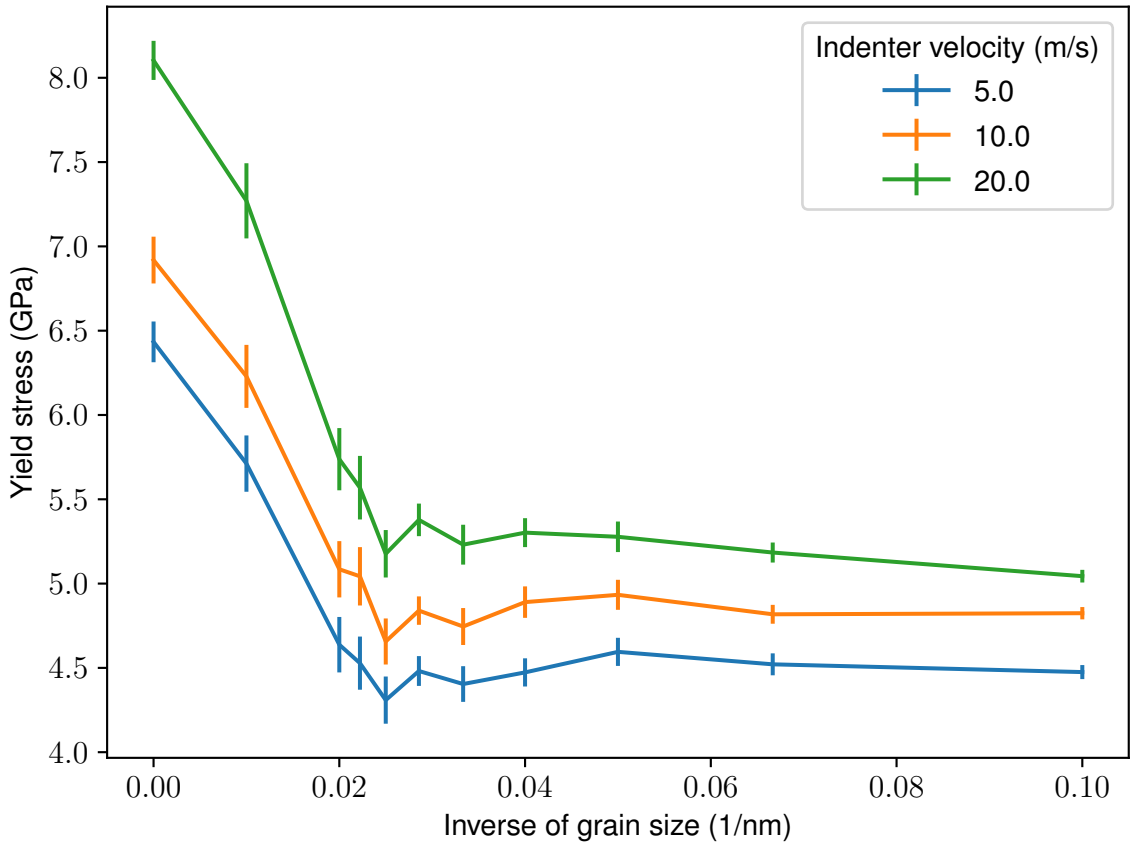


Figure 5.4. *Yield stress as a function of inverse of grain size for cone-shaped* 20 nm *midway diameter tantalum nanopillars, errorbars are SEM*

with the 1% offset method averaged over the results of the simulations of 40 different configurations of the same grain size compressed at the same strain rate. The errorbars are SEM. In the figure we can see that for smaller strain rates, the yield stress seems constantly to be lower than for higher strain rates. We can also see that for larger grains the yield stresses are higher than for smaller grain sizes. In the figure we can see that there seems to be some kind of transition for the yield stress as a function of grain size somewhere in the range of 30 nm and 50 nm. Below the transition zone the grain size does not affect the yield stress significantly. For grain sizes above that zone there seems to be a clear trend of higher yield stresses for larger grain sizes.

From the same figure it is notable to see the errorbars getting smaller for smaller grain sizes. This is most likely caused by the fact that there are more grains within the pillar if the grains are smaller. This results in averaging in the contribution from different grains and a single grain boundary doesn't affect the compression as much anymore.

We also get statistics of the global maxima of the stress-strain curves, which is illustrated in figure 5.5. For this approach we can notice that the general appearance of the figure is similar to figure 5.4, but the stress values seem to be higher. This is expected, due to the definition of global maximum, it is always greater or equal to the yield stress obtained

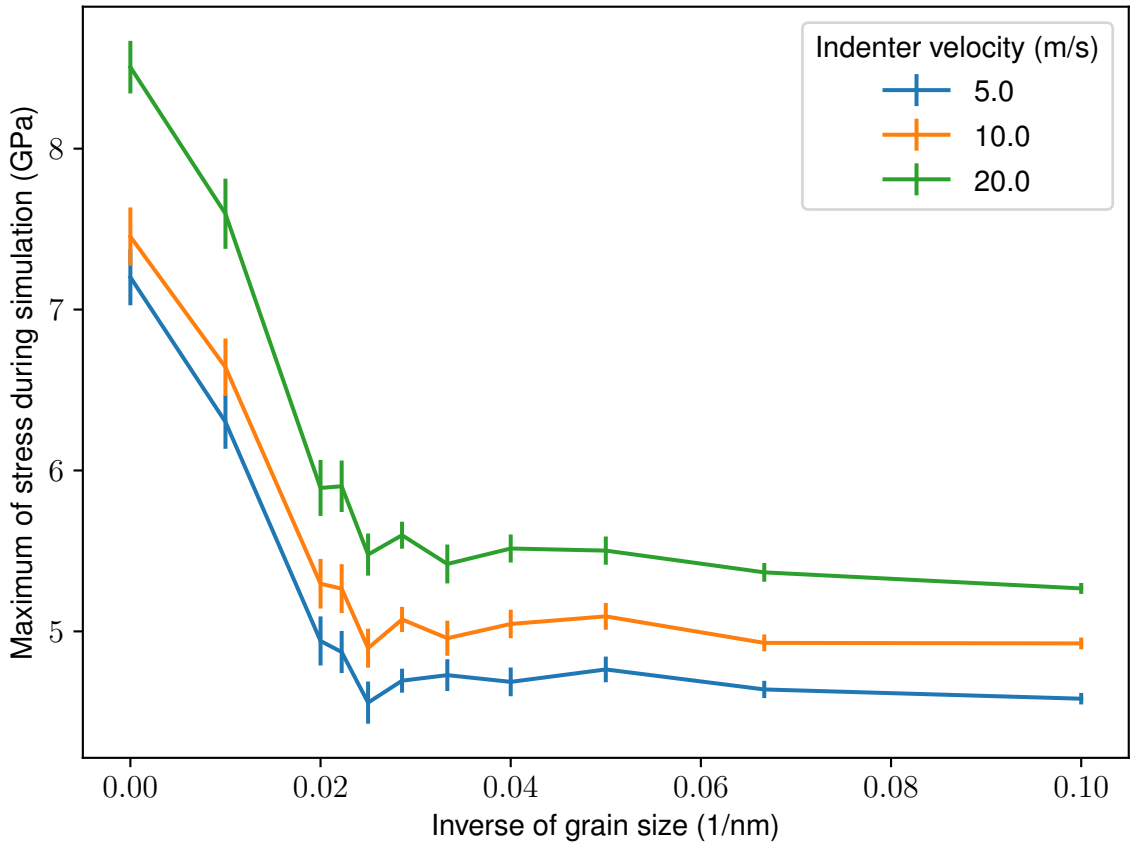


Figure 5.5. *Maximal stress along the stress-strain curve as a function of inverse of grain size for cone-shaped* 20 nm *midway diameter tantalum nanopillars, errorbars are SEM*

with 1.0% offset method.

With the data acquired, we can also try to extrapolate the yield stress to the $0\,\mathrm{m\,s^{-1}}$ indenter velocity matching the strain rate of $0\,\mathrm{s^{-1}}$. We can do this by considering the yield stress as a function of the indenter velocity for each of the grain sizes, and fit a linear function to those data. In figure 5.6 we can see the data represented in this way together with the fitted lines. Using the values of the yield stress extrapolated to the velocity of $0\,\mathrm{m\,s^{-1}}$ we can plot another figure. Figure 5.7 includes this data as a function of inverse of grain size. For the extrapolated values the shape of the curve resembles the finite strain rate figures for yield stresses. In addition to the general appearance being similar there now seems to be a minimum in the transition zone of the relation between the yield stress and the grain size. The transition zone of the relation between grain size and yield stress is still found between the grain size range of 30 nm and 50 nm. However, the value at the minimum has quite a large error, as indicated by the value of SEM, and the existence of the minimum is therefore arguable. With grain sizes larger than the transition zone the yield stress exhibits a non-monotonic behaviour as the function of the grain size. There seems to be some kind of local maximum of yield stress somewhere near the inverse grain size of 0.05, which corresponds to the grain size of 20 nm.

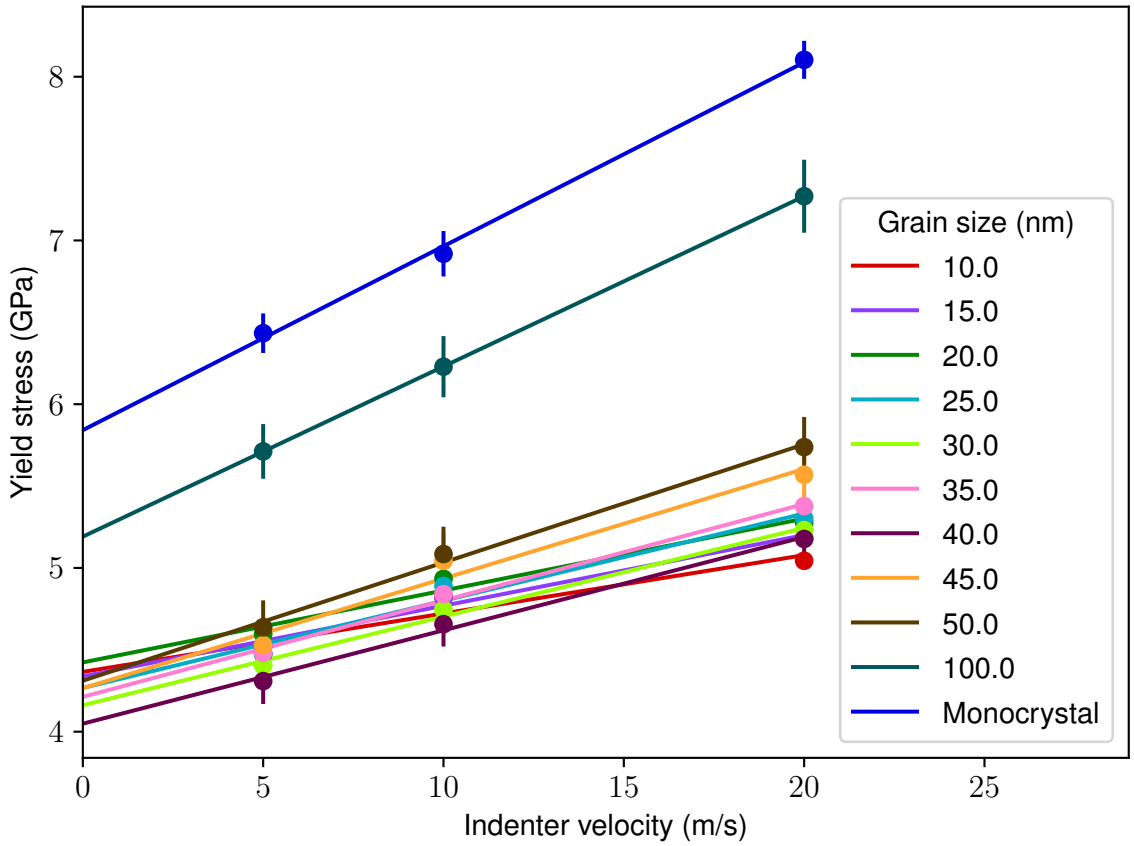


Figure 5.6. *Yield stress as a function of indenter velocity with fitted lines for cone-shaped* 20 nm *midway diameter tantalum nanopillars, errorbars are SEM*

5.2.4 20 m s^{-1} indenter plane velocity nanopillar statistics

A non-statistical study of the compression of tantalum nanopillars by means of molecular dynamics has already been made [8]. Simulations to compare with that study use a separate set of nanopillars (separate from subsection 5.2.3 data) that was also created for this thesis. That portion of the results uses same ranges for grain size, indenter velocity and pillar sizes as that study. In that study it was concluded that largest grain sizes resulted in significantly reduced yield stresses. This seems to be different from the results of the statistical analysis in this thesis. In my research (in figure 5.7) it was also possible to observe a drop in yield stress with larger grains, but the magnitude was not as big as in that study. Since the magnitude was so different the reasons for these drops in yield stresses are most likely different. It is important to keep in mind that the aforementioned study uses 2 fs timestep instead of 1 fs used in the simulations of this thesis. In addition the study uses a different method of pillar generation and different ensemble, namely NPT is used instead of NVT in this thesis. This being said, it seems that the study has only used 4 different bulk materials to create 16 different pillars. If there has been some kind of defect within the generated bulk material, it might have affected all the 4 pillars carved from that bulk material.

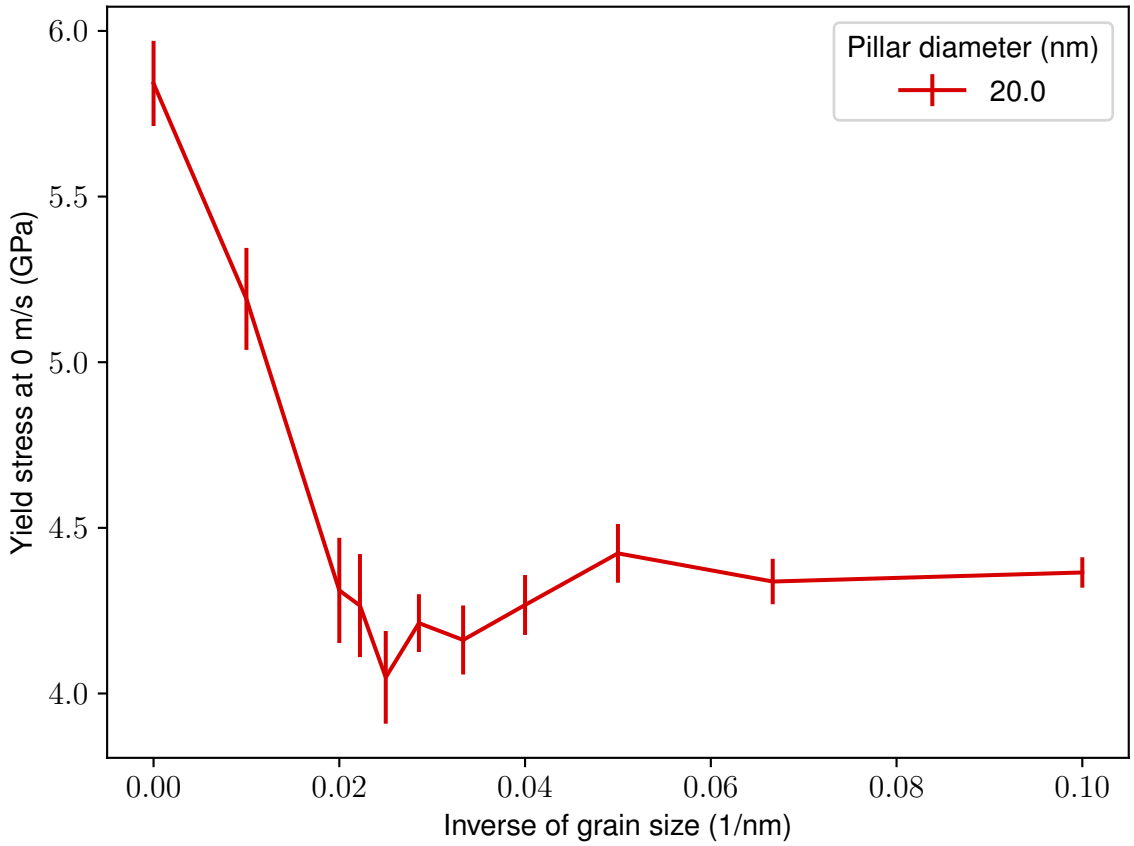


Figure 5.7. *Yield stresses extrapolated linearly to* $0\,\mathrm{m\,s^{-1}}$ *for cone-shaped* 20 nm *midway diameter tantalum nanopillars, points are means from extrapolating each sample separately to* $0\,\mathrm{m\,s^{-1}}$ *indenter velocity, errorbars are SEM from those means*

This thesis uses the statistical data extracted from a large number of simulations, one of these examples can be seen in figure 5.8 We can see that there seems to be a large amount of variation between different samples overall.

Furthermore this thesis uses SEM for the errorbars, the data is seen in figures 5.9, 5.10 and 5.11. From that data we can see that there seems to be not much correlation between the yield stress and the pillar diameter. But we can also see that for larger pillar diameters there seems to be a clear reduction in SEM. This could be explained with statistics. For larger grains there are fewer grains within a certain sized pillar which results in an orientation of a single grain, and thus the grain boundary, having greater effect on the compression stress. The contrary happens for smaller grains, there are multiple grain boundaries and a rotation or location of each individual grain do not have as great of an effect.

In figures 5.9 and 5.12 we observe a major discrepancy between the study and the results of the simulations in this thesis. The 50 nm grain size yield stresses are much lower in the study than in this thesis. There are a few possible reasons for the observed drop in yield stress for the study. Those reasons are discussed after the introduction of the other figures.

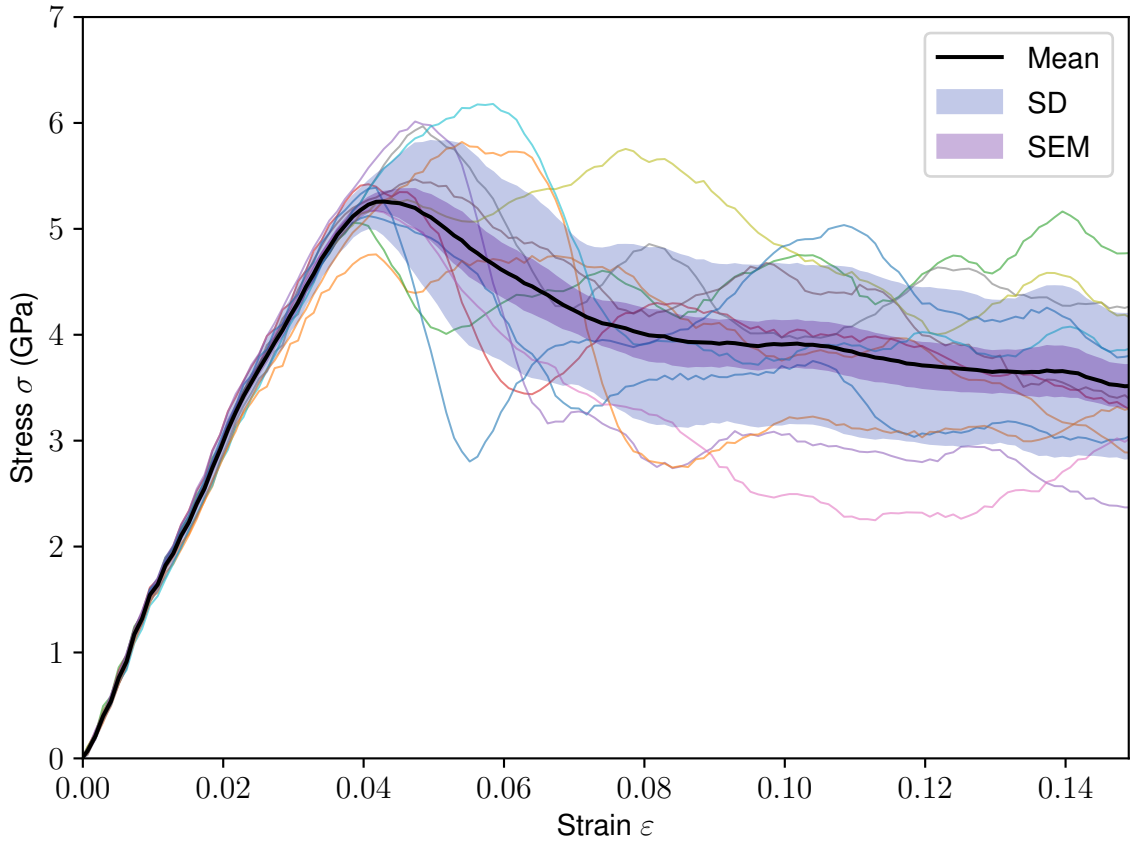


Figure 5.8. *Simulated stress strain curves of* 30 nm *average grain size* 30 nm *midway diameter nanopillars with* 20 m s^{-1} *indenter plane velocity*

In figure 5.10 we can see that there does not seem to be a clear way the grain size affects the yield stresses contrary to the other study in figure 5.13. It is also possible to see that there is a lot of error in this particular plot, and the smaller grain sizes seem to make the error smaller. This is again a result of single grain boundary factor being reduced due to there being more grain boundaries to begin with.

In figure 5.11 we can see this kind of averaging, resulting from larger number of grains within the pillar. This is visible in the greater errors for greater grain sizes, as well as greater errors for greater grain to pillar diameter ratios. It is also possible to note a slight increase in yield stress for the larger grain size to pillar diameter ratios. This similar slight increase of yield stress is also visible in the other study (illustrated in 5.14), but it is not clear in either of the studies.

As section 3.6.1 already concluded, there was a great probability, and there still is a slight possibility in Atomsk for voids to be created within a polycrystal. This might be one of the reasons the other study on Ta nanopillars sees a clear decrease in yield stress for the 50 nm grain sized pillars. If there has been a void within that specific polycrystal bulk material, all the pillars cut from the bulk material might, depending on the position of the void, have the void inside them. If this was the case, it could have contributed to the

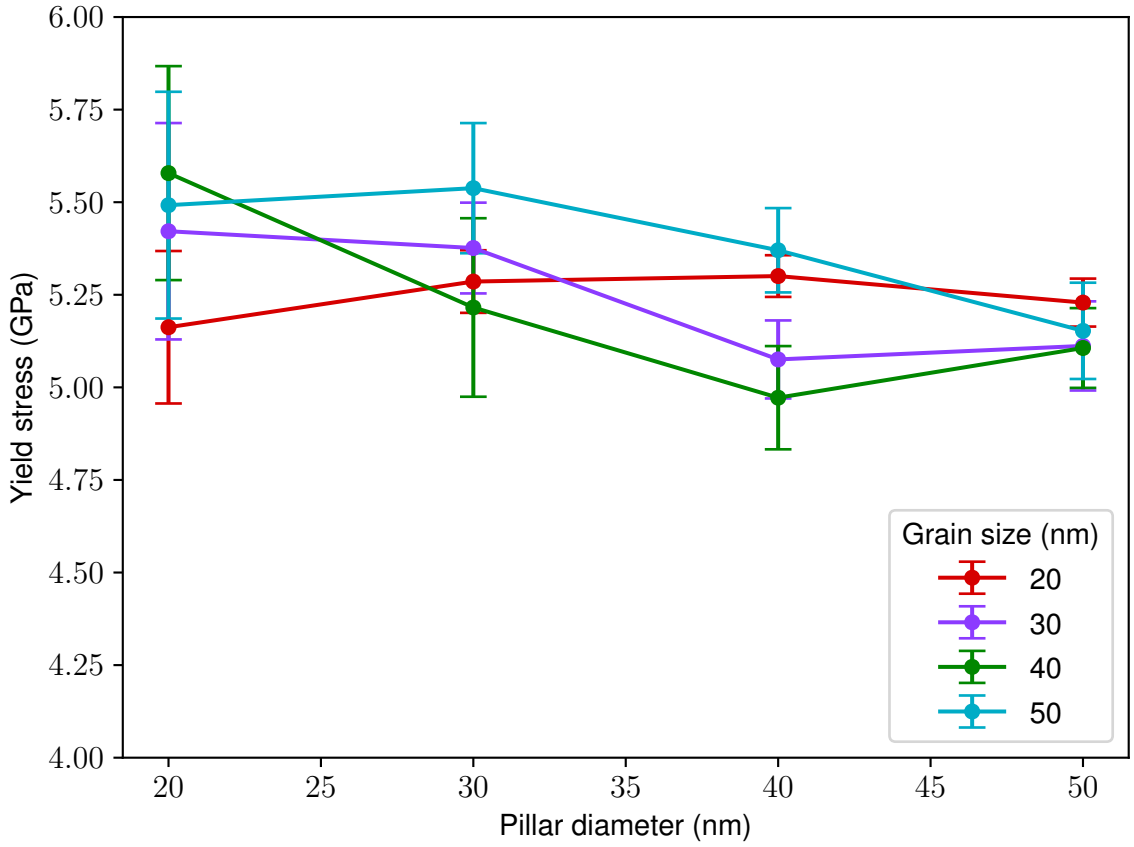


Figure 5.9. *Yield stress as a function of pillar diameter for cone-shaped tantalum nanopillars, errorbars are SEM*

significant drop of the yield stress reported in the publication.

In the study also the stress-strain curves for 20 nm and 50 nm grain size pillars were presented. Since the study is using the same method for extracting yield stresses from the stress-strain data, the values can be verified. By recalculating the yield stresses from the stress-strain figures, we get figures 5.15 and 5.16. If we take a look at these recalculations, it seems that despite the fact that the method for yield stress extraction should be the same, the stresses do not match with the recalculated ones. The recalculated stresses seem to be closer for 50 nm. The yield stresses differ so much that the values are outside the errorbars. If standard deviation was used, the values might be within the errorbars, but for example 20 nm the recalculated values differ on some points more than the original values.

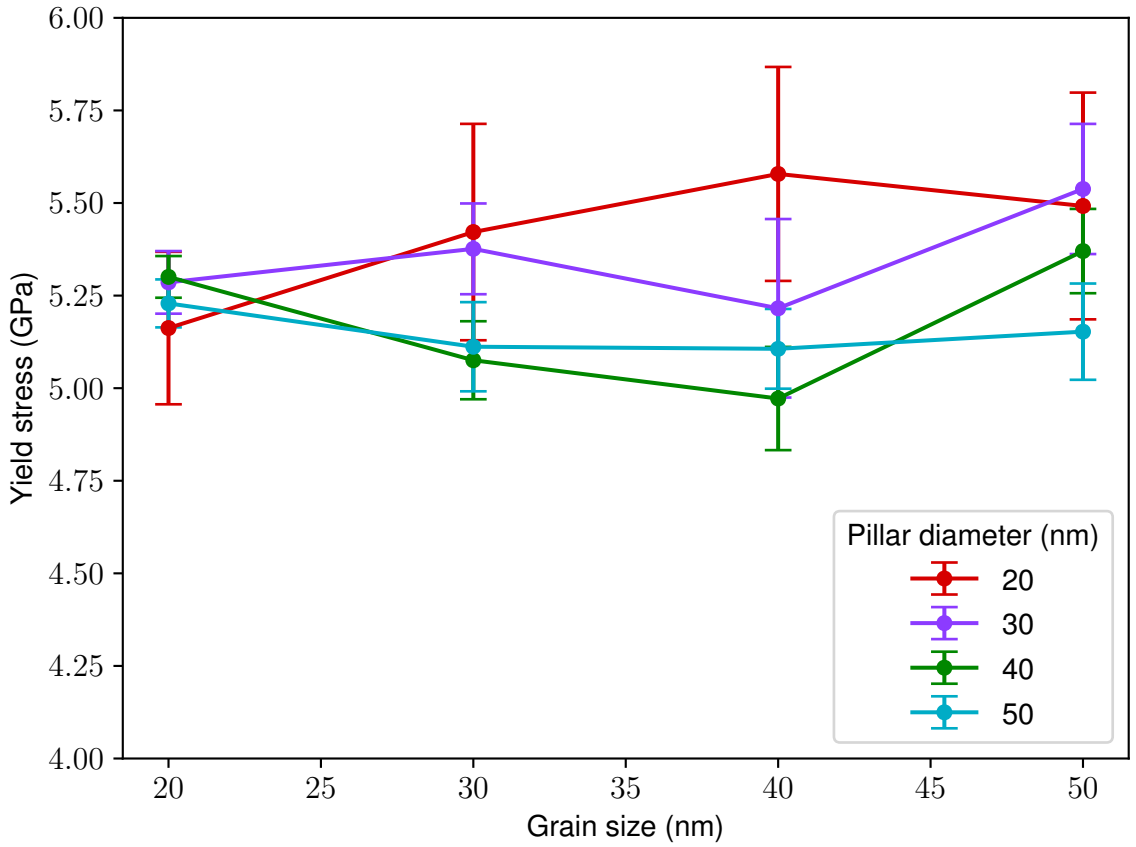


Figure 5.10. *Yield stress as a function of grain size for cone-shaped tantalum nanopillars, errorbars are SEM*

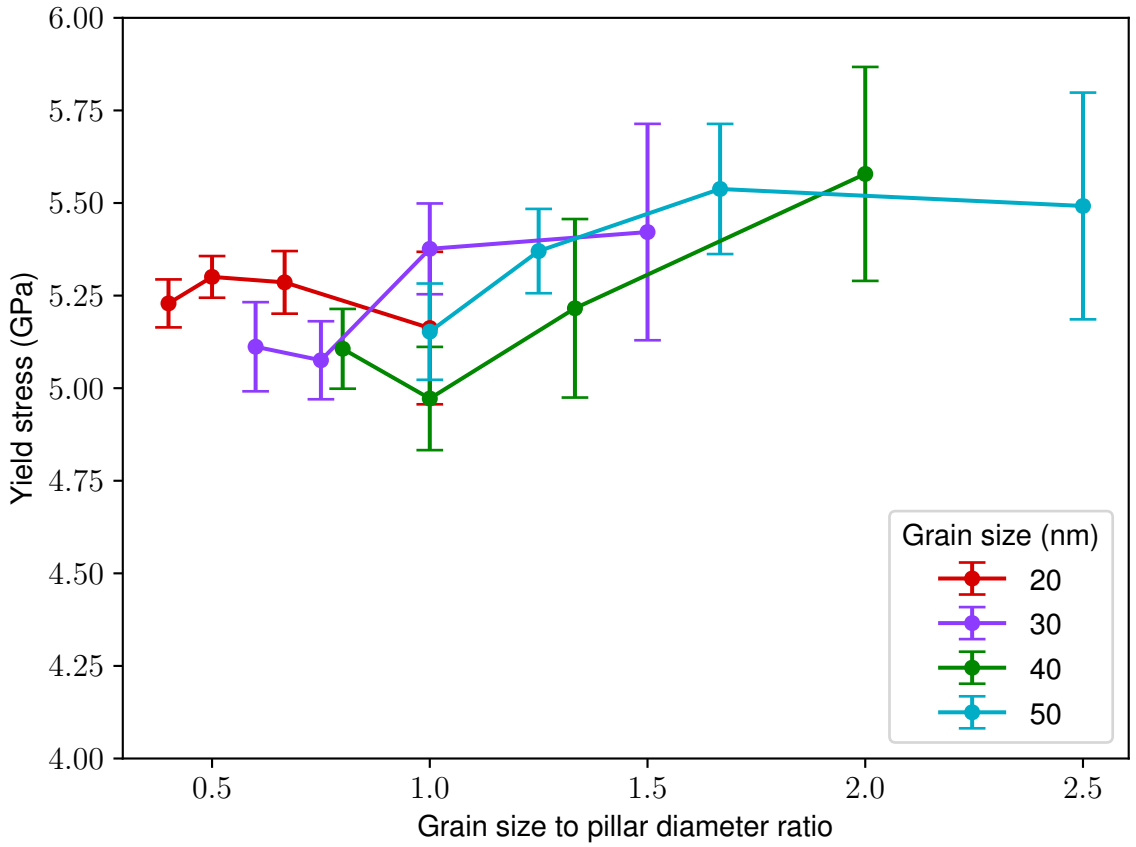


Figure 5.11. *Yield stress as a function of grain size divided by pillar diameter for cone-shaped tantalum nanopillars, errorbars are SEM*

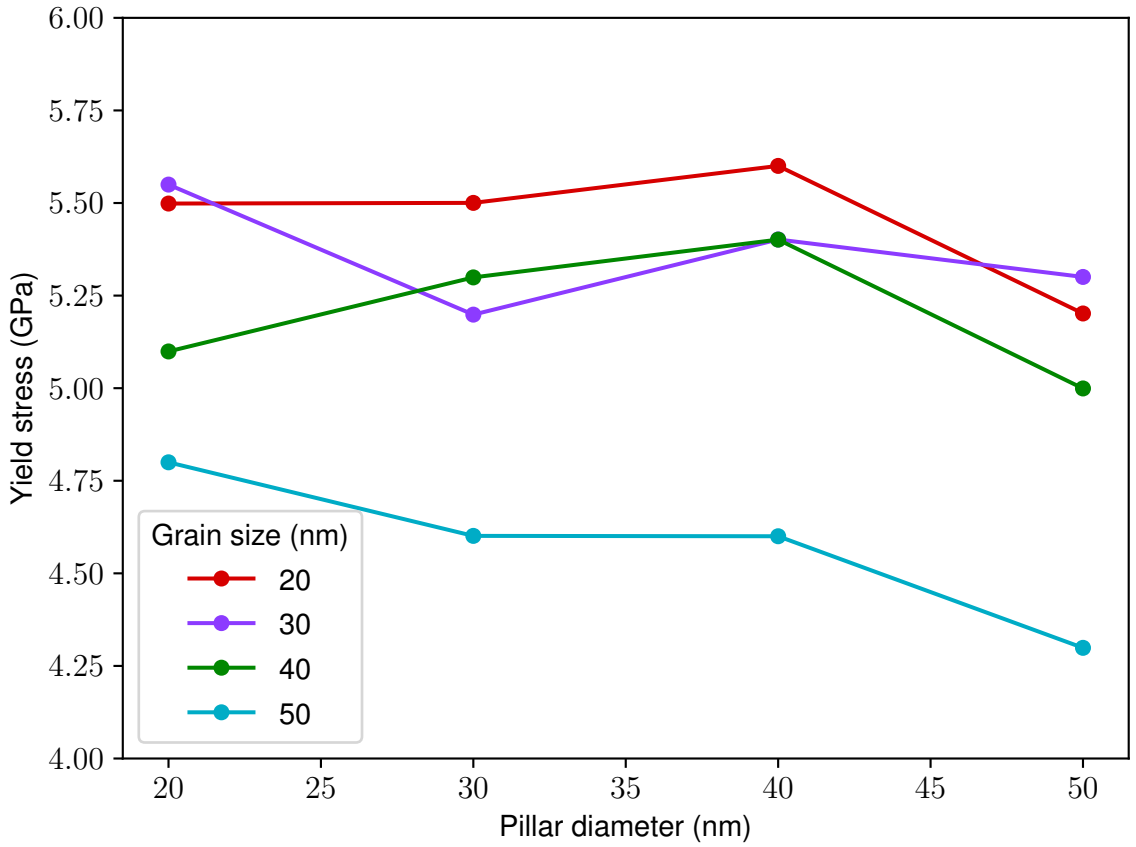


Figure 5.12. *Yield stress as a function of pillar diameter for cone-shaped tantalum nanopillars, data from a similar study [8]*

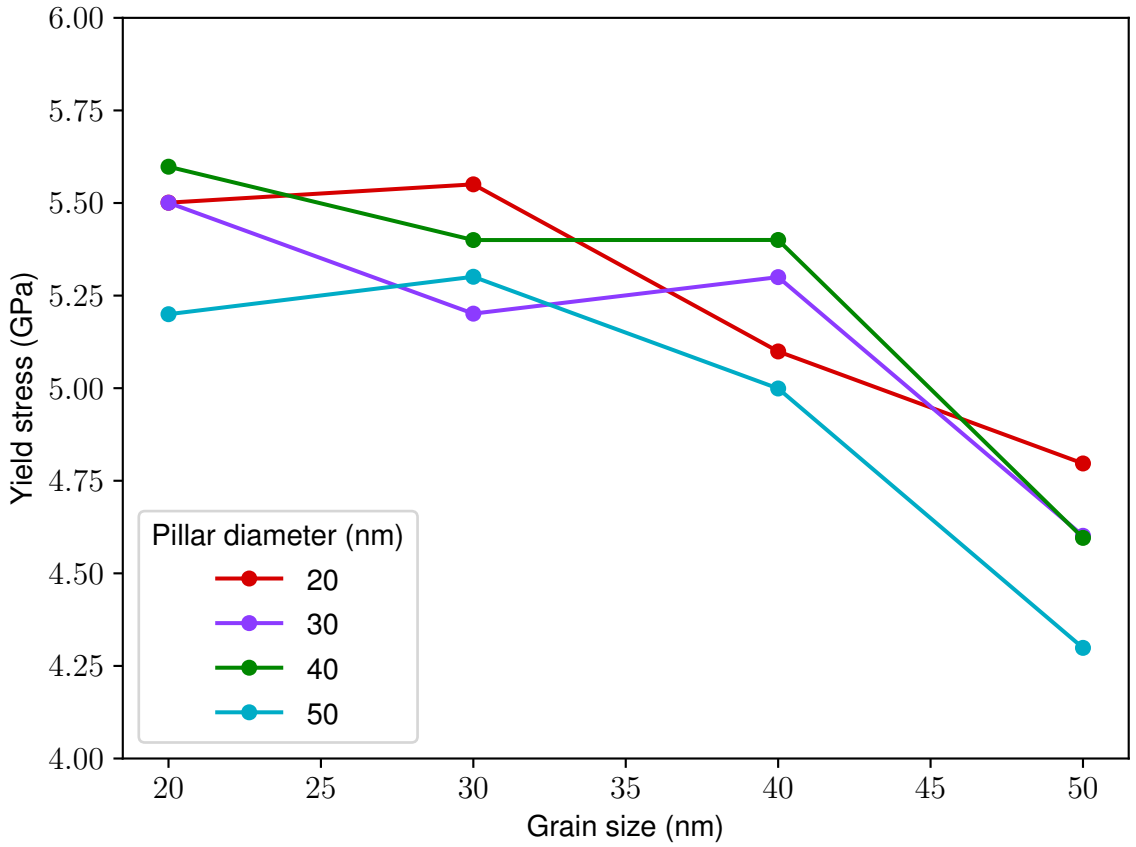


Figure 5.13. *Yield stress as a function of grain size for cone-shaped tantalum nanopillars, data from a similar study [8]*

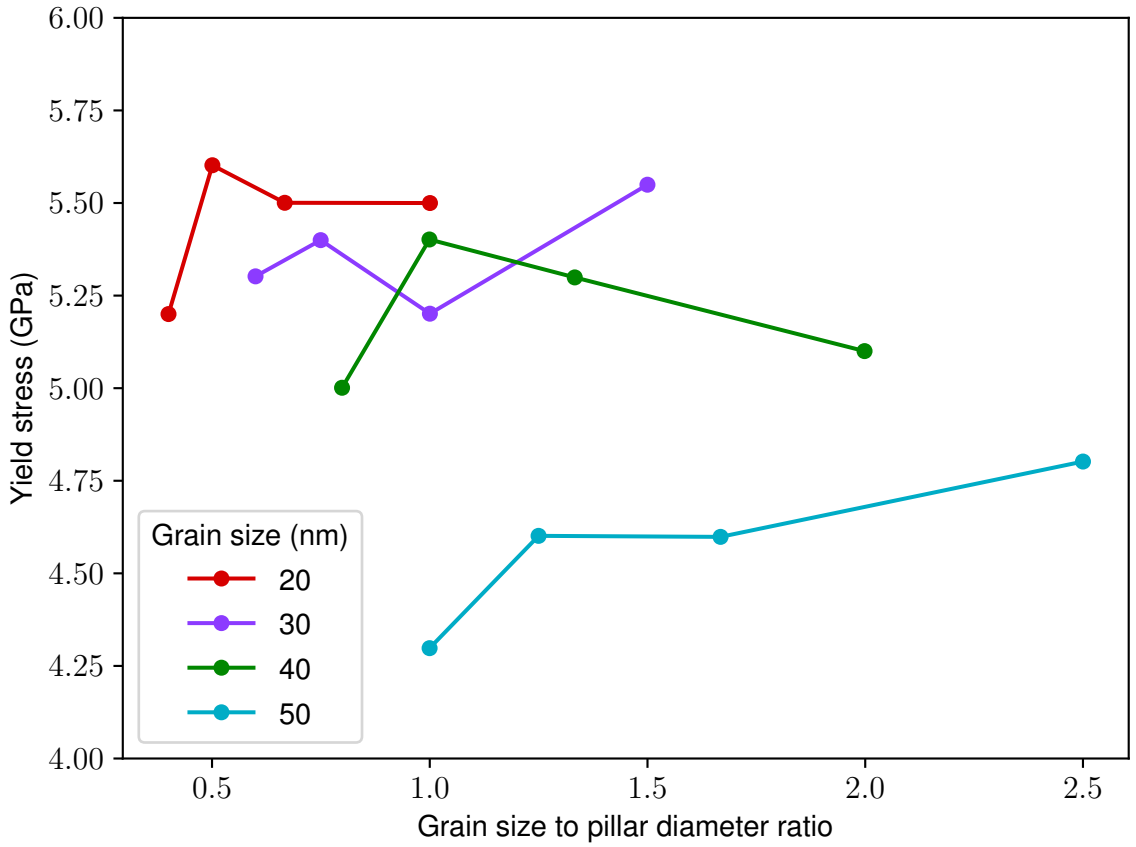


Figure 5.14. *Yield stress as a function of grain size divided by pillar diameter for cone-shaped tantalum nanopillars, data from a similar study [8]*

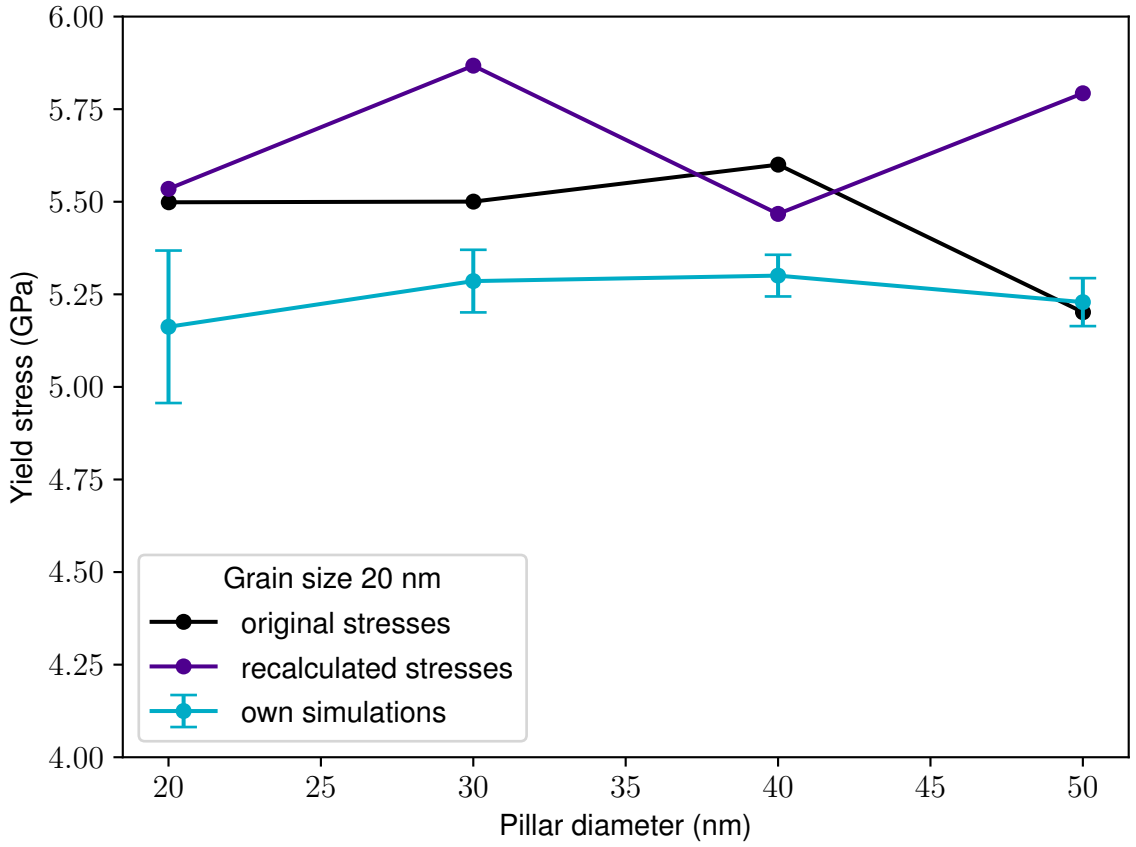


Figure 5.15. *Original and recalculated stresses for cone-shaped tantalum nanopillars from* 20 nm *grain size data from [8] with a comparison to this thesis' statistics, errorbars for this thesis' statistics are SEM*

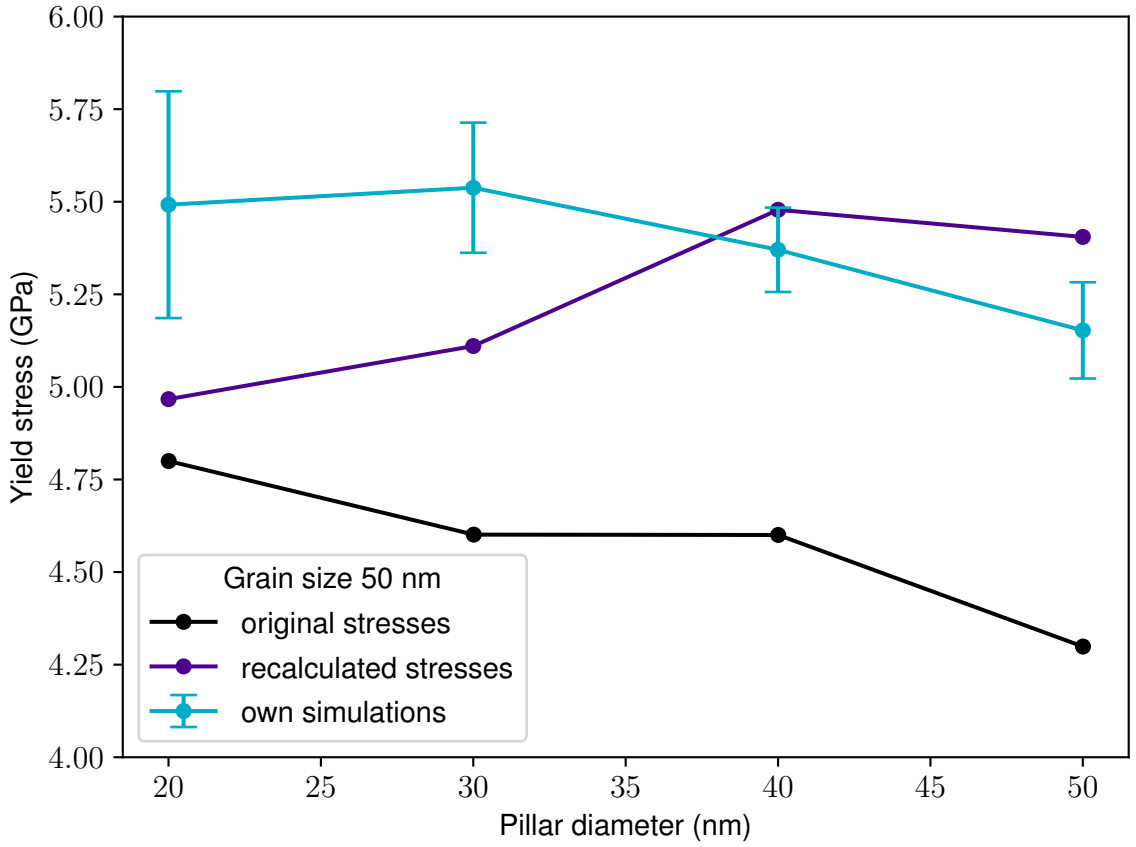


Figure 5.16. *Original and recalculated stresses for cone-shaped tantalum nanopillars from* 50 nm *grain size data from [8] with a comparison to this thesis' statistics, errorbars for this thesis' statistics are SEM*

6. CONCLUSIONS

The thesis studied the mechanical response and structure of nanopillars. Tantalum nanopillar compression was studied with molecular dynamics simulations performed with LAMMPS. A new program was created for the generation of initial configurations of nanopillars for molecular dynamics simulations. With the simulations it was possible to generate stress-strain data and that was analyzed statistically. An effort for comparing the results with existing ones was made, and a contradiction between those and the new results was found.

Due to the performance and functionality differences discussed in this thesis, Nanocrystal generator seems like a better choice for generation of polycrystalline configurations than Atomsk. This is true especially if one needs large nanopillars carved out of a huge bulk material with a lot of grains. In this case the performance and correctness of Nanocrystal generator seem to be better than those of Atomsk. For that reason Nanocrystal generator was a better choice for the purposes of this thesis where it was utilized to produce initial configurations of polycrystalline nanopillars.

This thesis studied the stress-strain response of tantalum nanopillars. The results consisted of two portions, one of which considered only 20 nm midway diameter pillars. The second part of the results contained stress-strain data from nanopillars of different sizes and that data was compared to another study with analogous data, which, however, considered only one sample for each of the sizes studied making the approach non-statistical.

In 20 nm midway diameter nanopillars there seems to be a transition between two regimes in the relation between grain size and yield stress. Yield stress seems to depend highly on the grain size, if the grain size is larger than 35 nm and as the pillar approaches the single grain (monocrystal) limit, the yield stress gets higher. For grain sizes smaller than 35 nm the change in the grain size does not have a clear effect on the yield stress. In addition a higher strain rate, or in this case a higher indenting plane velocity, increases the yield stress. For slower indenter velocities there appears to be a possible minimum point with a grain size of 35 nm, however due to errorbars being as large as they are, it is not yet possible to confirm if that indeed is a minimum.

In the second portion of the study, the statistical data of different sized nanopillars were compared with the study with non-statistical data. There seems to be a discrepancy

between the results between those two studies. Since statistical data shows also the errorbars in the case of this thesis, the research can be assumed more accurate. The yield stresses were also calculated from the original stress-strain curves from the non-statistical study, and it was found that they do not match the values shown in the figures in the manuscript. This indicates a contradiction between the two studies. In this thesis there was no clear drop in yield stress for any of the grain sizes and pillar sizes. This is yet another point where one of the most important conclusions of this thesis comes up, statistical analysis is required to create representative data which is necessary to draw proper conclusions.

There are a lot of possibilities for extending the current research. For the transition grain size of two regimes in 20 nm, it would be an interesting study to see what happens if the pillar size is changed. Does the transition point move in some direction, and if it does, which direction? It would also be interesting to try to check if that minimum indeed exists. It would also be interesting to check that also with experiments with nanoindenters. For all of the results in this thesis, the range for all the possible parameters could be increased. There is a possibility for simulating with even smaller strain rates, larger and or smaller pillars and with larger or smaller grain sizes. It is also interesting to take a note on the experiments where the indenting plane is stopped [15], and test those by means of simulations.

REFERENCES

[1] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton. "LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales". In: *Computer Physics Communications* 271 (2022), p. 108171. URL: https://www.sciencedirect.com/science/article/pii/S0010465521002836.

[2] A. Stukowski. "Visualization and analysis of atomistic simulation data with OVITO-the Open Visualization Tool". In: *MODELLING AND SIMULATION IN MATERIALS SCIENCE AND ENGINEERING* 18.1 (Jan. 2010).

[3] M. Sarvilahti, A. Skaugen, and L. Laurson. "Machine learning depinning of dislocation pileups". In: *APL Materials* 8.10 (2020), p. 101109. URL: https://doi.org/10.1063/5.0020376.

[4] Y. Cheng and E. Ma. "Atomic-level structure and structure–property relationship in metallic glasses". In: *Progress in Materials Science* 56.4 (2011), pp. 379–473. URL: https://www.sciencedirect.com/science/article/pii/S0079642510000691.

[5] M. Delincé, Y. Bréchet, J. Embury, M. Geers, P. Jacques, and T. Pardoen. "Structure–property optimization of ultrafine-grained dual-phase steels using a microstructure-based strain hardening model". In: *Acta Materialia* 55.7 (2007), pp. 2337–2350. URL: https://www.sciencedirect.com/science/article/pii/S1359645406008391.

[6] Q. Liu, A. Roy, and V. V. Silberschmidt. "Size-dependent crystal plasticity: From micro-pillar compression to bending". In: *Mechanics of Materials* 100 (2016), pp. 31–40. URL: https://www.sciencedirect.com/science/article/pii/S0167663616300722.

[7] Y. Liu, S. Kondo, H. Yu, K. Yabuuchi, and R. Kasada. "Statistical Approach for Understanding the Effect of Specimen Size on the Yield Stress and Its Scattering in Mechanically-Alloyed Cu and ODS-Cu Obtained by Micro-Pillar Compression Test". In: *MATERIALS TRANSACTIONS* 61.5 (2020), pp. 955–962.

[8] W. Yang, C. J. Ruestes, Z. Li, O. T. Abad, T. G. Langdon, B. Heiland, M. Koch, E. Arzt, and M. A. Meyers. "Micro-mechanical response of ultrafine grain and nanocrystalline tantalum". In: *Journal of Materials Research and Technology* 12 (2021), pp. 1804–1815. URL: https://www.sciencedirect.com/science/article/pii/S2238785421003070.

[9] N. Kheradmand, H. Vehoff, and A. Barnoush. "An insight into the role of the grain boundary in plastic deformation by means of a bicrystalline pillar compression test and atomistic simulation". In: *Acta Materialia* 61.19 (2013), pp. 7454–7465. URL: https://www.sciencedirect.com/science/article/pii/S1359645413006629.

[10] M. D. Uchic, D. M. Dimiduk, J. N. Florando, and W. D. Nix. “Sample Dimensions Influence Strength and Crystal Plasticity”. In: *Science* 305.5686 (2004), pp. 986–989. URL: https://www.science.org/doi/abs/10.1126/science.1098993.

[11] D. M. Dimiduk, C. Woodward, R. LeSar, and M. D. Uchic. “Scale-Free Intermittent Flow in Crystal Plasticity”. In: *Science* 312.5777 (2006), pp. 1188–1190. URL: https://www.science.org/doi/abs/10.1126/science.1123889.

[12] P. Hirel. “Atomsk: A tool for manipulating and converting atomic data files”. In: *Computer Physics Communications* 197 (2015), pp. 212–219. URL: https://www.sciencedirect.com/science/article/pii/S0010465515002817.

[13] C. Kittel. *Introduction to solid state physics Eighth edition*. Ed. by S. Johnson. John Wiley & Sons, Inc, 2005.

[14] C. A. Volkert and A. M. Minor. “Focused Ion Beam Microscopy and Micromachining”. In: *MRS Bulletin* 32.5 (2007), pp. 389–399.

[15] G. Mohanty, J. Wehrs, B. L. Boyce, A. Taylor, M. Hasegawa, L. Philippe, and J. Michler. “Room temperature stress relaxation in nanocrystalline Ni measured by micropillar compression and miniature tension”. In: *Journal of Materials Research* 31.8 (Apr. 2016), pp. 1085–1095. URL: https://doi.org/10.1557/jmr.2016.101.

[16] G. Mohanty, J. M. Wheeler, R. Raghavan, J. Wehrs, M. Hasegawa, S. Mischler, L. Philippe, and J. Michler. “Elevated temperature, strain rate jump microcompression of nanocrystalline nickel”. eng. In: *Philosophical magazine (Abingdon, England)* 95.16-18 (2015), pp. 1878–1895.

[17] M. Griebel. *Numerical Simulation in Molecular Dynamics Numerics, Algorithms, Parallelization, Applications*. eng. 1st ed. 2007. Texts in Computational Science and Engineering, 5. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[18] *Multiscale Modelling of Plasticity and Fracture by Means of Dislocation Mechanics*. eng. 1st ed. 2010. CISM International Centre for Mechanical Sciences, Courses and Lectures, 522. Vienna: Springer Vienna, 2010.

[19] K. Binder. *Monte Carlo Simulation in Statistical Physics An Introduction*. eng. 5th ed. 2010. Graduate Texts in Physics. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[20] E. Voimanen. *Nanocrystal Generator*. 2022. URL: https://gitlab.com/EricssonWilli/nanocrystal-generator.

[21] H. Schreiner. *CLI11: Command line parser for C++11*. Version 2.2.0. Dec. 2022. URL: https://github.com/CLIUtils/CLI11.

[22] G. Guennebaud, B. Jacob, et al. *Eigen v3*. 2010. URL: http://eigen.tuxfamily.org.

[23] C. H. Rycroft. *Voro++*. 2008. URL: https://math.lbl.gov/voro++/.

[24] *Lammps data format description*. URL: https://docs.lammps.org/2001/data_format.html.

[25] P. M. Larsen, S. Schmidt, and J. Schiøtz. “Robust structural identification via polyhedral template matching”. In: *Modelling and Simulation in Materials Science and*

Engineering 24.5 (May 2016), p. 055007. URL: https://dx.doi.org/10.1088/0965-0393/24/5/055007.

[26] A. Gaschler. *3D Rotation Converter*. URL: https://www.andre-gaschler.com/rotationconverter/ (visited on 2022-11-28).

[27] A. Gaschler. *3D Rotation Converter GitHub*. URL: https://github.com/gaschler/rotationconverter (visited on 2022-11-28).

[28] N. I. of Standards and Technology. *NIST Periodic Table*. 2019. URL: https://www.nist.gov/pml/periodic-table-elements.

[29] M. Matsumoto and T. Nishimura. "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator". eng. In: *ACM transactions on modeling and computer simulation* 8.1 (1998), pp. 3–30.

[30] *LAMMPS Publications*. URL: https://www.lammps.org/papers.html.

[31] X. D. Dai, Y. Kong, J. H. Li, and B. X. Liu. "Extended Finnis–Sinclair potential for bcc and fcc metals and alloys". In: *Journal of Physics: Condensed Matter* 18.19 (Apr. 2006), p. 4527. URL: https://doi.org/10.1088/0953-8984/18/19/008.

[32] *LAMMPS Documentation, atom_style command*. URL: https://docs.lammps.org/atom_style.html (visited on 2022-12-19).

[33] *CSC Documentation, Puhti*. URL: https://docs.csc.fi/computing/systems-puhti/ (visited on 2022-12-19).

[34] *CSC Documentation, Available batch job partitions*. URL: https://docs.csc.fi/computing/running/batch-job-partitions/ (visited on 2022-12-19).

[35] A. Stukowski. *Crystal Analysis Tool*. 2012. URL: https://gitlab.com/stuko/crystal-analysis-tool.

[36] A. Stukowski, V. V. Bulatov, and A. Arsenlis. "Automated identification and indexing of dislocations in crystal interfaces". In: *Modelling and Simulation in Materials Science and Engineering* 20.8 (Oct. 2012), p. 085007. URL: https://doi.org/10.1088/0965-0393/20/8/085007.

[37] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95.

[38] M. L. Waskom. "seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60 (2021), p. 3021. URL: https://doi.org/10.21105/joss.03021.

[39] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. URL: https://doi.org/10.1038/s41586-020-2649-2.

[40] T. pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. URL: https://doi.org/10.5281/zenodo.3509134.

[41] W. McKinney. “Data Structures for Statistical Computing in Python”. In: *Proceedings of the 9th Python in Science Conference*. Ed. by S. van der Walt and J. Millman. 2010, pp. 56–61.

APPENDIX A: INPUT SCRIPT FOR NANOPILLAR COMPRESSION IN LAMMPS

```
units           metal
boundary        m m m
atom_style      atomic

read_data       ../pillars/ta20_20_0.lmp

pair_style      eam/alloy
pair_coeff * * ../ta.efs Ta

variable VOL            equal "vol"
variable zHeight        equal 605.4

velocity        all create 300.0 42 rot yes

minimize        1.0e-6 1.0e-8 100 1000

# fix two layers so that they do not move
# group for two layers
region          bottom_two_layers block 0 800 0 800 0 6.6052
group           bottom_two_layers region bottom_two_layers
# group for all the other atoms
group others subtract all bottom_two_layers
# fix
fix             BASE bottom_two_layers setforce 0 0 0
velocity        bottom_two_layers set 0 0 0

# relaxing
timestep 0.001
reset_timestep 0
```

```
compute         KE all ke

fix             1 others nvt temp 300.0 300.0 0.01
fix             TD all ave/time 5000 1 5000 c_thermo_temp &
                c_thermo_press c_thermo_pe c_KE v_VOL c_thermo_press[3] &
                file "td_eq.dat" mode scalar format "%24.16E"

thermo          5000

# dump            1 all atom 5000 dump.NG.eq

run             40000

# disable fixes and dumps for the following part
unfix           TD
# undump          1



# and deformation

reset_timestep 0

compute stressPerAtom others stress/atom NULL

compute p_all others reduce sum c_stressPerAtom[1] c_stressPerAtom[2] &
        c_stressPerAtom[3]

variable planeZ equal "v_zHeight - 0.0002*step"
variable strain equal "(v_zHeight - 5.4 - v_planeZ)/(v_zHeight - 5.4)"
variable stressPress equal -(c_p_all[3])/ &
         ((v_planeZ-6.6052)*31415.926535897932)*1E-4

fix             INDENTER others indent 10 plane z v_planeZ hi

fix             TD others ave/time 5000 1 5000 c_thermo_temp &
                c_thermo_press c_thermo_pe c_KE v_VOL &
                file "td.dat" mode scalar format "%24.16E"
```

```
fix             STRESS others ave/time 5000 1 5000 c_thermo_temp &
                c_thermo_press[3] v_VOL v_strain v_stressPress &
                file "stress.dat" mode scalar format "%24.16E"

# dump            2 all atom 5000 dump.NG.deform

restart         477000 restart.*.compress
run             477000
```