2016-06-01

Non-Schmid Effects and Criteria for Dislocation Nucleation on Different Slip Systems at Grain Boundaries

Richard Durtschi Wyman
Brigham Young University

Non-Schmid Effects and Criteria for Dislocation Nucleation on Different Slip Systems at Grain Boundaries

Richard Durtschi Wyman

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Eric R. Homer, Chair
David T. Fullwood
Oliver K. Johnson

Department of Mechanical Engineering

Brigham Young University

June 2016

ABSTRACT

Non-Schmid Effects and Criteria for Dislocation Nucleation
on Different Slip Systems at Grain Boundaries

Richard Durtschi Wyman
Department of Mechanical Engineering, BYU
Master of Science

Criteria for grain boundary dislocation nucleation are developed. A bicrystal containing two grain boundaries is placed under varying triaxial stress states using molecular dynamics. The local resolved shear, normal, and co-slip stresses needed for grain boundary dislocation nucleation are found. A framework is developed to detect the slip system grain boundary dislocation nucleation occurs on. A survey of the different ways grain boundary dislocation nucleation occurs in the sample shows a single grain boundary can nucleate dislocations in a rich variety of ways. Using the nucleation system and resolved stress values, criteria for grain boundary dislocation nucleation on different slip systems are developed. The proposed form of nucleation criterion suggests the activation stress has a linear dependence one the resolved shear, normal, and co-slip stresses. A residual analysis largely validates the efficacy of the proposed linear model. We show that the nucleation slip system cannot be predicted by a maximum Schmid factor analysis due to the non-Schmid resolved normal and co-slip terms. We show that a system's global pressure generally fails to predict nucleation; a local stress in the grain being nucleated into should be used. Using the nucleation criteria for each slip system, a yield surface for dislocation nucleation is built for the grain boundary used in this work.

Keywords: grain boundaries, grain boundary dislocation nucleation, plasticity, triaxial stress, molecular dynamics

ACKNOWLEDGMENTS

So many people have helped me get to this point in my life. I thank my parents for raising me to be inquisitive, curious, and educationally minded. I thank my undergraduate advisors, Doctor Vanfleet and Doctor Davis, for preparing me for graduate school. I thank Levi Morrison of the Fulton Super Computing lab for helping me compile and use software for this project. In short, I thank the countless people who have been a positive influence in my life. I thank my committee members, Doctor Fullwood and Doctor Johnson, for giving their time to provide comments and constructive criticism that have greatly enhanced this paper. Particular thanks is owed to Doctor Eric Homer. He has provided me excellent learning opportunities. He has given me enough rope to learn but kept it short enough that I don't hang myself. Doctor Homer has provided me a world class education.

This work could not have happened without the help of the U.S. Department of Energy, Office of Science, Basic Energy Sciences under Award #DE-SC0012587. This great country's continued funding of sciences pays the wage of some, inspires many, and benefits all. May research continue and expand.

I'm thankful for Brigham Young University. They have set me up to succeed in life. I'm thankful for the Church that has subsidized my education as well as taught me correct principles. Most of all, I'm thankful for our Savior who stands at the head of this Church. When I returned to graduate school, I came for a Masters degree; I found something even better. Meeting Cindee was the best thing that ever happened to me. Cindee, thank you for saying yes. I love you. (Oh, and thanks for the copy edits too!)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

Almost all consumer metal products are polycrystalline metal. Polycrystalline metal consists of small crystals known as grains that are fused together. The seams between the grains are known as grain boundaries (GBs). Grain boundaries are extremely important in governing material strength, ductility, and many other material properties.

This thesis is mainly focused on a particular type of GB plasticity mechanism known as Grain Boundary Dislocation Nucleation in which a dislocation (a 1D crystalline defect [1]) grows from a GBs surface to relieve local stress. This section will explain, in some depth, the relevant physics and governing principles of grains, GBs, and dislocations that serve as background to the main topic. It will also cover some of the mathematical and computational tools that were used to perform the research.

1.1 Dislocations

Dislocations are 1D irregularities in crystals. They are the primary unit of plasticity in single crystal systems. Dislocations possess stress fields whose interactions lead to work hardening [2]. This work is concerned primarily with a particular subset of dislocations known as partial dislocations which are the main way GB nucleate dislocations [3–5]. In a partial dislocation, atoms exhibit a lattice distortion (quantified by a Burger's vector) that does not restore the crystalline stacking sequence [6]. Partial dislocations are always accompanied by a two-dimensional crystalline defect which, in this work, is a stacking fault in the crystalline material.

When a partial dislocation nucleates from a GB, it leaves behind a stacking fault. The stacking fault breaks the stacking sequence in a perfect crystal. For instance, in FCC metals, atomic planes are stacked on top of each other following an *ABCABC*... pattern; however, a stacking fault will create a stacking sequence like *ABC***ABA***BCABC*... [2]. The *ABA* stacking fault is a small

region where an HCP stacking sequence occurs. We will use these stacking faults to determine the types of partial dislocations a GB nucleates.

1.2 Grain Boundaries

Grain boundaries form the divisions between grains. They have a complex structure that generally positions the atoms such that the net potential energy of all atoms is minimized. At one point in time, the atoms composing a GB were believed to be amorphous; however, it is now known that the GB has a complex atomic structure.

The GB energy is calculated by drawing a box around a GB and summing the potential energy of all the atoms in the box. The cohesive energy (potential energy of an atom in a perfect crystal) of each atom is then subtracted. Finally, the energy value is divided by the area of the GB in the box. This leaves a value of energy per unit GB area. This value is always positive for single element systems (and generally positive for alloys, unless the structure attacts solute atoms in a way that drammatically lowers the energy), which is not surprising as GBs are high energy structures. Measuring the evolution of the GB energy relative to the system energy can show when GB dislocation nucleation occurs [7].

Grain boundaries can be specified by the orientation of their surrounding grains relative to each other. Grain boundaries are categorized by their coincidence site lattice (CSL) number [8]. This number represents the mismatch between two grains by quantifying the number of atoms that would lie on top of each other if the two grains were superimposed on top of each other. It is always an odd number. A CSL value of 1 would correspond to a perfect lattice with no GB in it.

1.3 Atomistic Modeling

Atomistic modeling is a computational method that allows researchers to model systems of atoms. Atomistic modeling is less accurate than methods such as density functional theory that account for electron interactions but more accurate than finite element methods that do not model atoms and rely on constitutive models to account for any atomic level effects.

Molecular statics is an atomistic technique that can measure the energy of a system. It can also be used to take a high energy system and move atoms around until a local energy minimum

is found. By considering thousands of slightly different atomic configurations containing a grain boundary and using molecular statics to find the system's local energy minimum, a physically realistic GB can be developed. This technique was employed by another reseracher to build the grain boundary system used in this research [9, 10]. In this project, molecular statics is used to convert a only partially periodic system into a fully periodic system.

Molecular dynamics is an atomistic technique in which atoms move under Newtonian physics. The method tracks the position and velocity of each atom. A simulation consists of multiple time steps. For each time step, the acceleration on each atom is calculated by finding the force on each atom (using a potential the same way as done in molecular statics) and then using Newton's 1st law $F = Ma$ to find the acceleration. While specific techniques vary, the program modifies the velocities on each atom based on the acceleration and then changes the position of each atom based on its velocity. The process is then repeated over and over again, each step moving the system forward in time. More sophisticated implementations (such as Velocity-Verlet [11]) use a half time step and ensure the system is time reversible [12, Sec 9.3.1]. Molecular dynamics will be used in this project to simulate grain boundary dislocation nucleation.

1.3.1 Potentials

An integral part of atomistic modeling is a potential that allows forces on atoms to be calculated. The potential allows the determination of an atom's potential energy based on its surroundings. By taking the gradient of the potential energy, a force acting on an atom can be obtained [12, Sec 5.8.2].

Potentials can be fit to experimental data (the Lennard-Jones [13] potential is fit against experimentally determined bond length and strength), developed using quantum mechanics (the ReaxFF force field potential is fit using quantum chemistry [14]), or using combinations of the two. Care must be taken when choosing a potential. Potentials are developed to fit specific material properties and may fail to reproduce phenomena a user desires to model [15] [12, Sec 5.8.2]. The nickel potential used in this research was fit against the stacking fault energy of nickel [16]. The stacking fault energy (as well as other parameters) is known to affect dislocation slip [17] meaning this potential should work well for the current research. The potential has been used in other grain boundary dislocation nucleation research [18].

While atomistics can use many types of potentials to determine the forces acting on the atoms, perhaps the most popular potential used for modeling metal systems is the embedded atom model (EAM) potential [19]. The EAM potential computes the energy of each atom by considering an atom's neighbors as well as an embedding term which models the density of the surrounding electrons (as captured by the density of atoms) [20]. An EAM potential is used in this work to model Nickel.

1.3.2 LAMMPS

LAMMPS (Large-Scale Atomic/Molecular Massively Parallel Simulator) is the molecular dynamics software used in this project [21]. It is developed by Sandia National Laboratories and has found wide use in the computational materials science committee. It is released under the GNU public license making it free for educational and commercial use, although restrictions are placed on commercializing the software itself. The program is under active development and is available from `http://lammps.sandia.gov/`.

LAMMPS is written in C++ and is heavily optimized using neighbor lists [12, Sec 6.4.1] to reduce an $O(N^2)$ problem to an $O(N)$ problem (at least for the EAM potential). It comes with a wide suite of built in metrics and is very easy to add new features to. For instance, we added code to compute the slip vector [22] metric for this project. The program is highly modular and can run on conventional hardware or graphical processing unit (GPU) accelerated hardware [23–25].

1.3.3 Super Computing

Given the computationally expensive nature of atomistic simulations, molecular dynamics simulations performed in this work were performed on Brigham Young University's super computer [26]. The super computer allows a user to submit jobs to a scheduler [27]. An individual job is submitted as a shell script. The super computer will put the job into a queue. When resources become available for the job, the scheduler will allocate the resources and start the job. When the job finishes, the resources will be returned and the scheduler will use them for another job. This process allows thousands of jobs to be queued or running at the same time.

A typical super computing work flow for this project consisted of a shell script submitting hundreds of jobs with slightly different parameters to the job scheduler. Each of these jobs would make an output folder directory and then run a LAMMPS stress simulation. After all simulations completed, the results could be copied to a local computer for post processing.

1.4 Grain Boundary Dislocation Nucleation

Grain boundaries (GBs) play an important role in strength and can govern material properties as grain size shrinks to nano-crystalline dimensions [28–31]. GBs are barriers to dislocation motion, and act as sources and sinks for dislocations. Additionally, GBs can accommodate stress through GB sliding, GB void growth, and GB migration. Of primary interest in this work is the phenomenon of GB dislocation nucleation, which is a form of heterogeneous dislocation nucleation where a GB relieves local stress by emitting dislocations in one or both of the grains it separates. The GB dislocation nucleation is generally a non-reversible plastic mechanism that permanently deforms the material with few exceptions. The nucleation event results in a drop in GB interfacial energy [7, 18] and stress relaxation [32–35].

Much like homogeneous dislocation nucleation in the absence of a GB [36–38], dislocation nucleation at a GB is known to have non-Schmid dependence [32, 39]. In particular, the critical resolved shear stress, τ_{crss}, for dislocation nucleation is also a function of resolved normal stress, σ_{rns}, which is the stress normal to the dislocation's slip plane, as shown in Figure 1.1. When the resolved normal stress is compressive, it is harder for atoms to glide over each other; when tensile, it is easier for atoms to glide. Although appearing to be less important, the resolved co-slip shear stress, τ_{rco}, which is the shear stress perpendicular to the slip plane normal and slip direction as shown in Figure 1.1, may also affect dislocation nucleation from GBs [32].

Several theories have been proposed to predict the conditions associated with dislocation nucleation. Spearot, et. al. investigated the phenomenon by applying uniaxial tension to a number of $\langle 100 \rangle$ and $\langle 110 \rangle$ tilt GBs [32]. By using a least squares fit, they built an equation to predict the uniaxial stress required to cause GB dislocation nucleation. They found that dislocation nucleation from $\langle 100 \rangle$ tilt GBs is governed primarily by the resolved shear stress while $\langle 110 \rangle$ tilt GBs are governed primarily by resolved normal stress. Additional work by Spearot et al. found that increasing nanoporosity in a GB led to nucleation at relatively low stresses [40]. Beyerlein, et. al. proposed

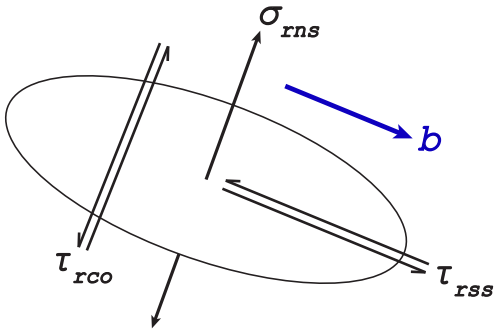


Figure 1.1: Schematic illustration of the stress components τ_{rss}, σ_{rns}, τ_{rco} on a dislocation with Burger's vector $\vec{b}$

that dislocation nucleation is favored on slip systems well aligned with intrinsic dislocations inside the GB. The likelihood of nucleation on some slip system is then proportional to the driving stress multiplied by a structure factor quantifying the favorability of some slip system [33]. Sangid, et. al. developed a method of measuring energy barriers to dislocation nucleation and found that the energy barrier for nucleation is inversely proportional to the static GB energy [18]. The current consensus is that GB dislocation nucleation is tied to GB structure and that resolved shear alone cannot predict when nucleation will occur.

Other factors surrounding the moment and location of GB dislocation nucleation have also received attention. Wu, et. al. showed that the point of nucleation within a GB is correlated with high Von Mises stress [34]. Similarly, Zhang, et. al. showed that the point of nucleation tends to occur at points with the most intense interface distortion [41]. Burbery, et. al. correlated the nucleation point with atoms whose virial stress has a large component normal to the GB and noted that per atom potential energy fails to predict the location of nucleation [42]. None of these observations are mutually exclusive as all suggest nucleation occurs at points of high stress.

In the present work, we examine a single GB under many different triaxial stress states. This approach enables us to better understand how different stress states influence dislocation nucleation on different slip systems. Using the stress states measured near the GB at the time nucleation occurs, we are able to determine the relative influence of shear, normal, and co-slip stresses. The analysis produces unique nucleation criteria for different slip systems that we combine to give an effective yield surface for dislocation nucleation.

CHAPTER 2. METHODOLOGY

2.1 Bicrystal Sample

The present work is focused on simulating an atomistic model of a Nickel GB subjected to a range of triaxial stress states and measuring the response on GB dislocation nucleation. The GB of interest is a $\Sigma 21b$ $[211]$ 44.415° symmetric twist GB and is one of two GBs in a fully periodic bicrystal containing two grains of differing widths. The bicrystal is shown in Figure 2.1 with atoms colored by their common neighbor analysis [43] value. The smaller middle grain, labeled grain A, is 126 Å wide, the larger outer grain, labeled grain B, is 234 Å wide. Note that since the GB is periodic, the two side regions are part of the same grain, namely grain B. The dimensions in the y and z directions are 33 Å and 32 Å respectively. In total, the system contains 31,500 atoms. The bicrystal is derived from one of the GBs prepared by Olmsted, et. al. [9, 10], which is made fully periodic by copying the grain boundary, flipping the copy upside down, and merging the two GBs into a bicrystal with additional matrix material. Molecular dynamics is performed with LAMMPS [21] utilizing GPU accelerated hardware [23–25]. The Foiles-Hoyt Nickel embedded atom potential [16] is used throughout this work. Visualization of the bicrystal is performed with OVITO [44]. The GB is simulated at 0.1 Kelvin to minimize the contributions of thermal energy, as has been done in other work [39]. The GB is equilibrated at 0.1 Kelvin and zero pressure for 100 ps prior to subjecting the GB to any mechanical loading.

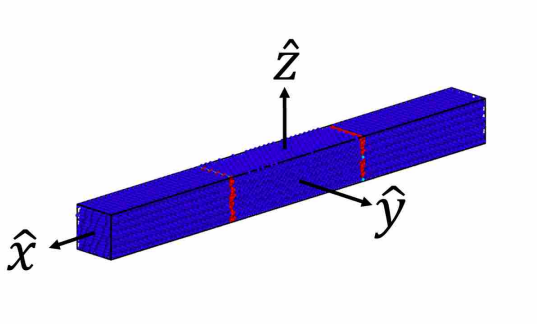


Figure 2.1: The $\Sigma 21b$ [211] 44.415° symmetric twist GB bicrystal used in this work. The grain boundaries are shown in red, FCC matrix material is shown in blue

2.2 Triaxial Stress States

To induce GB dislocation nucleation, the bicrystal will be placed under many different triaxial stress states. These triaxial stress states are specified by a unit vector, λ, with each component of the vector representing a relative stress magnitude in the $\hat{x}$, $\hat{y}$, and $\hat{z}$ dimensions. For instance, the unit vector $\lambda = \begin{pmatrix} 1 & -2 & 3 \end{pmatrix} / \sqrt{14}$ would represent a triaxial state where the normal stress in the $\hat{x}$ dimension would be tensile, the normal stress in the $\hat{y}$ dimension would be compressive and double the magnitude of the normal stress in the $\hat{x}$ direction, and the normal stress in the $\hat{z}$ dimension would be tensile and triple the magnitude of the normal stress in the $\hat{x}$ direction.

The λ vectors are chosen by picking 386 points approximately equidistributed about a unit sphere [45]. The Cartesian coordinates of each vertex become the λ unit vectors. Points were chosen by starting with a cube and subdividing each face into four squares. Each vertex is then projected onto a unit sphere [45]. The subdivision process is repeated a total of three times. The progression of the subdivision/projection method is shown in Figure 2.2 with the 386 vertices (which became the 386 λ vectors) shown in the final sub figure.

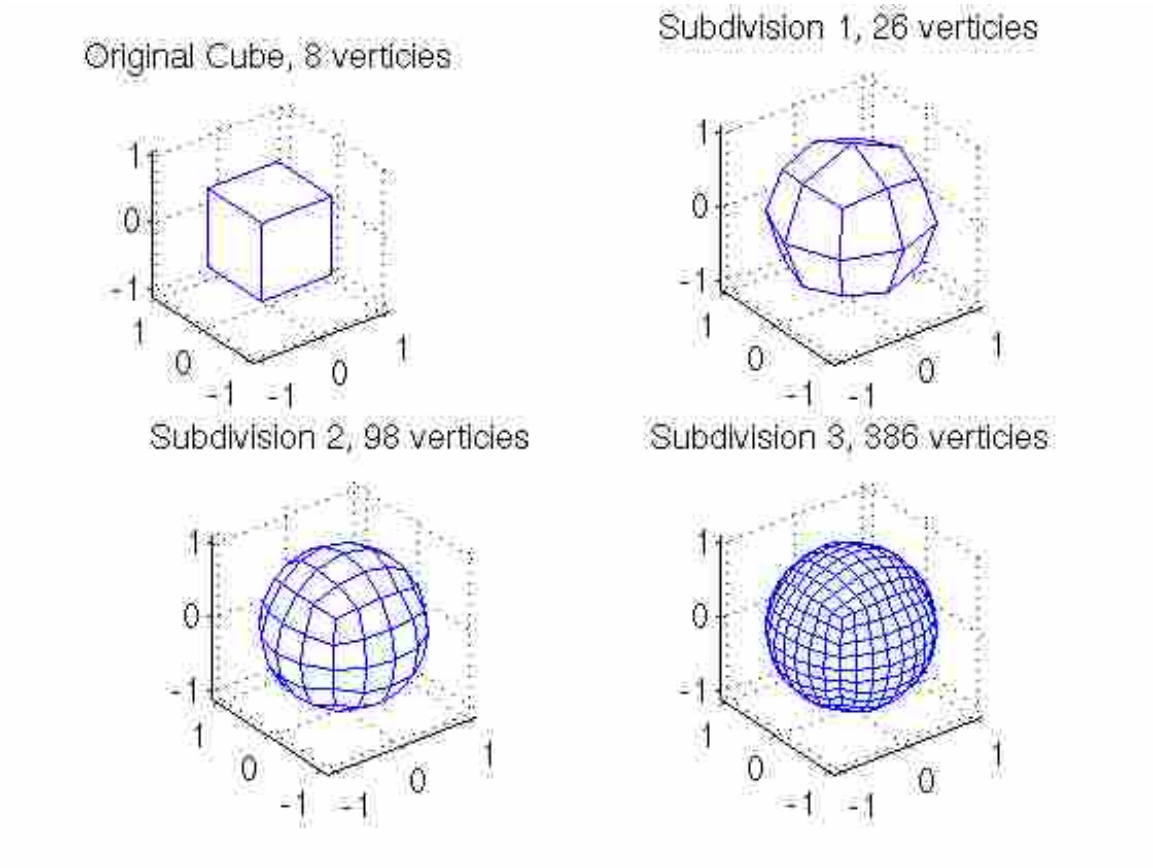


Figure 2.2: The subdivision process to choose triaxial stress states. Each vertex is a λ vector

2.3 Simulation

For each triaxial stress state, molecular dynamics is used to simulate the GB with the intention of inducing GB dislocation nucleation. The Nose-Hoover NPT ensemble is used to maintain the temperature at 0.1 Kelvin and ramp the applied pressure in the three orthogonal directions from 0 GPa while attempting to keep the relative applied pressures in conformance with the simulation's λ vector. The ramp rate of the applied pressures is such that the geometric norm of the applied stresses ($\sqrt{\sigma_x^2+\sigma_y^2+\sigma_z^2}$) increases at a rate of 100 MPa/ps. The simulation is terminated shortly after GB dislocation nucleation occurs.

Since the simulation cell is forced to maintain orthogonality, the shear stresses are not barostatted and are non-zero. Thus, while only orthogonal normal stresses are applied to the bicrystal, these stresses are not in a principal stress state reference frame. This is problematic because it means shears are applied on the GB which could lead to undesirable GB sliding or mi-

gration. However, in this work, the conditions surrounding GB dislocation nucleation are analyzed in terms of the local stresses near the GB rather than the global pressure. As a result, even though the bicrystal is not placed under true triaxial stress, the applied stress still meets the primary objective to place the GB under many different stress states. In practice, almost all of the simulations exhibited the GB dislocation nucleation mechanism.

2.4 Post Processing

2.4.1 Measurement of Grain Boundary Dislocation Nucleation Slip System

For the Nickel GB analyzed here, partial dislocations are the primary dislocation observed, so dislocation nucleation is analyzed in terms of the twelve $\{111\}\langle 11\bar{2}\rangle$ partial dislocation slip systems. A partial dislocation leaves a stacking fault behind it [46]; this stacking fault is analyzed to determine the slip system of the partial dislocation. Partial dislocations, instead of full dislocations, are expected in short atomistic experiments using Nickel because of the ratio between Nickel's unstable and stable stacking fault energies [47]. When a full dislocation occurs, it is always via a partial dislocation being emitted, followed by a stacking fault, followed by a trailing partial dislocation. Such full dislocations are analyzed as partial dislocations by looking at the intermediate stacking fault in the same way done with partial dislocations.

For each snap shot of the simulation, the slip system that nucleated is obtained by analyzing the atoms in stacking faults, which are identified as those atoms having a non-FCC structure, as obtained by common neighbor analysis [43]. Atoms are grouped into what we term colonies: a seed atom is added to a new group and atoms within a specified cutoff distance of the seed atom who have a similar slip vector [22] are admitted to the group. Each of these atoms is then analyzed for neighbors with similar slip vectors (previously admitted atoms are not revisited). This process is carried out recursively on all admitted atoms until no more atoms can be added to the group. This group is then termed a colony. A new seed atom is picked and the process continues until all stacking fault atoms have been added to a colony.

A slip system is assigned to each colony by fitting a normal vector to the atoms in the colony: of the four possible partial slip plane normal vectors, the slip plane normal vector that is most aligned with the fit vector is chosen as the slip plane normal vector of the colony. A slip

direction for the colony is found by averaging the slip vectors of every atom in the colony: of the three possible partial slip directions, the slip direction that is most aligned with the colony slip direction becomes the slip direction of the colony. The slip plane normal and slip direction are then combined to form a slip system for the colony. In some cases, the software reported an unrealistic slip system in which the dot product between the independently determined slip plane normal and slip direction is not zero. This generally occurs on very small colonies of only a couple atoms and are computational artifacts. Such colonies make up an insignificant portion of the entire stacking fault content of a simulation and are ignored.

Once a slip system has been assigned to each colony, conglomerated slip system content is calculated by adding the number of atoms for each colony that nucleated on the same slip system. The slip system with the largest number of stacking fault atoms on it is chosen as the primary GB dislocation nucleation slip system for that particular snap shot in that grain.

After this process has been repeated on each snap shot in a simulation, the slip system that the simulation nucleated on is determined by considering each snap shot in turn. It is common for a simulation to have multiple nucleation events; this research is only concerned with the first nucleation event (see [7] for precedent). Subsequent nucleation events are in a heavily deformed bicrystal and are outside the scope of this paper. Although only the first nucleation event is considered, many simulations nucleate on two slip systems simultaneously; in these cases, both slip systems are considered as nucleation slip systems for the simulation.

The above process is repeated for both grains for each simulation resulting in an index of what type of grain boundary dislocation nucleation each simulation experiences. Fourteen and ten simulations in grain A and grain B respectively do not exhibit categorizable grain boundary dislocation nucleation from the GB of interest. These simulations are dropped from further analysis.

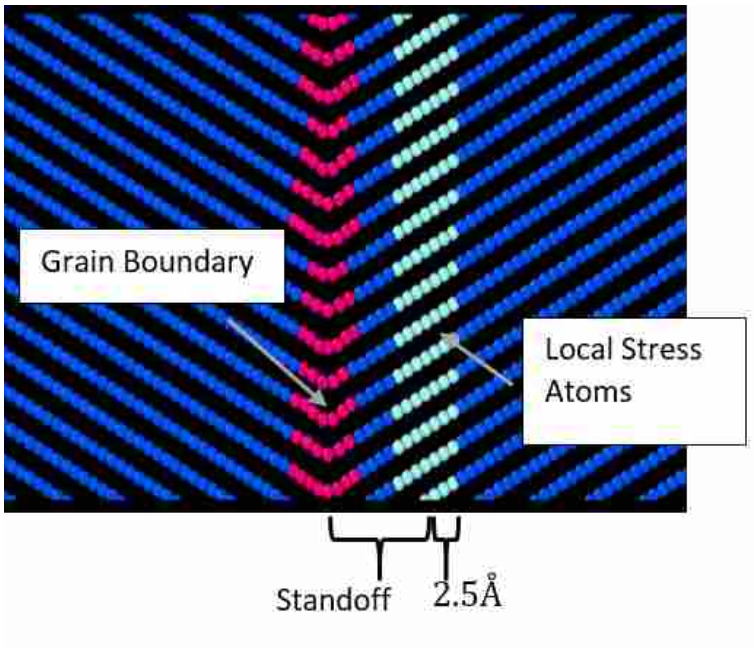
2.4.2 Local Stress

To determine the conditions that lead to GB dislocation nucleation on a particular slip system, the GB local stress is found as a function of time. The local stress is considered to be of greater interest than the global bicrystal pressure because the GB exhibits a back stress. This back stress alters the stress state under which GB dislocation nucleation is initiated and this back stress is only captured by a local stress definition. The local stress is calculated by averaging the virial

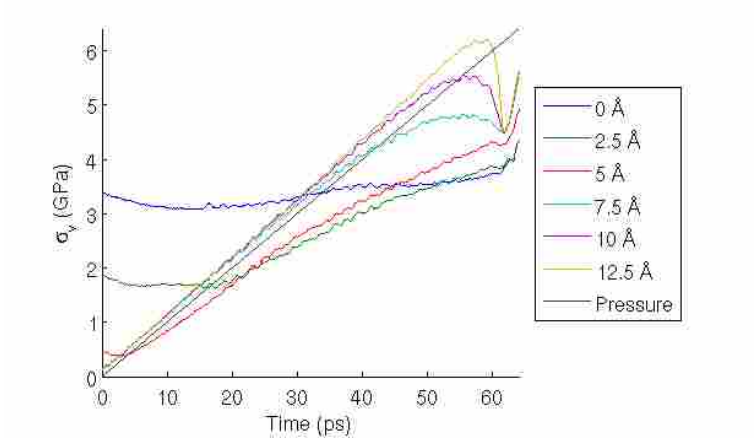
stress [48] of all atoms within 2.5 Å of the position X Å away from the middle GB where X is what we term the standoff distance.

Choosing the correct standoff distance requires a balance between two competing factors: a standoff too close to the GB is heavily influenced by the disordered structure of the GB [49, 50] and will make the measurement of the stress challenging. A distance too far from the GB will measure a stress different than that actually causing the dislocation to nucleate. Standoff distances ranging from 0 to 12.5 Å are considered as well as the global system pressure. The width of atoms is always 5 Å wide (2.5 Å on either side of the standoff), of which the group of atoms for a standoff of 7.5 Å is illustrated in Figure 2.3a.

The volume averaged von Mises stress for the various standoff distances is plotted in Figure 2.3b as a function of time for a uniaxial tensile simulation ($\lambda = [1, 0, 0]$). When nucleation occurs, we expect a sudden drop in the von Mises stress as the GB dislocation nucleation mechanism accommodates the built up stress. It can be seen that for the 0 Å standoff, the stress inside the GB starts off high because of the GBs inherently high stress. As the system is placed under higher stress, the local stress doesn't evolve significantly even though the global stress is ramped up (as indicated by the slightly increasing pressure). The 2.5 Å case starts at a somewhat reduced stress but also still has a significant initial value, presumably from the GB atoms in the group. By the 5 Å standoff, the von Mises stress ramps from near zero and reaches a maximum before dropping at the point of nucleation, which drop could be mistaken for noise in the curve. The 7.5 Å standoff is similar though the stress drop at nucleation is now clearly a departure from the loading. The 10 Å and 12.5 Å standoffs appear to overshoot the maximum stress, because they are so far offset from the point of nucleation that they exceed the stress while waiting for the dislocation to arrive. The global pressure is insensitive to the nucleation event because it considers all the atoms in the simulation. We take the 7.5 Å standoff as the ideal local stress definition because it is insensitive to initial GB stress and captures the GB dislocation nucleation event. Stresses reported from this point on will be calculated using the local 7.5 Å stress definition in the grain of interest.



(a)



(b)

Figure 2.3: (a) The local stress in a grain is captured by averaging the virial stress of all atoms within a 2.5 Å window of a particular standoff from the center of the GB

(b) Von Mises stress as a function of time for different standoffs in a uniaxial tensile test. The 7.5 Å standoff is the closest to the GB that is not affected by the GB core

2.4.3 Resolved Stress Calculations

Foundational to this work is the ability to calculate resolved stresses on different slip systems. Consider the slip system

$$(n_x n_y n_z)[d_x d_y d_z] \tag{2.1}$$

where n is the slip plane normal and d is the slip direction. We define $v = cross(n,d)$ which is a vector perpendicular to the slip plane normal and slip direction. We define a stress in its own frame as

$$\sigma = \begin{bmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{xy} & \sigma_{yy} & \tau_{yz} \\ \tau_{xz} & \tau_{yz} & \sigma_{zz} \end{bmatrix} \tag{2.2}$$

To compute the resolved stress components of σ we need to find a transformation matrix that changes from the stress frame into the slip system's frame. The stress frame's axes expressed in the lab frame are $x_{stress} = [1,0,0]$, $y_{stress} = [0,1,0]$, and $z_{stress} = [0,0,1]$. The slip system's axes as expressed in the lab reference frame are $x_{slip} = [n_x n_y n_z]$, $y_{slip} = [v_x v_y v_z]$, and $z_{slip} = [d_x d_y d_z]$. The transformation matrix, T, between the stress frame and the slip system frame is then

$$T = \begin{bmatrix} n_x & n_y & n_z \\ v_x & v_y & v_z \\ d_x & d_y & d_z \end{bmatrix} \tag{2.3}$$

Once T is known, we can transform the stress in the stress frame into the slip system frame by computing $\sigma_R = T * \sigma * T'$ where σ_R is the resolved stress. The resolved shear stress, τ_{rss}, is component $\sigma_R(1,3)$. The resolved normal stress, σ_{rns}, is component $\sigma_R(3,3)$. The resolved co-slip stress, τ_{rco}, is component $\sigma_R(2,3)$. By computing $\sigma_R = T * \sigma * T'$, we find the resolved shear stress to be

$$\begin{aligned} \tau_{rss} = {} & n_x * (d_x * \sigma_{xx} + d_y * \sigma_{xy} + d_z * \sigma_{xz}) \\ & + n_y * (d_x * \sigma_{xy} + d_y * \sigma_{yy} + d_z * \sigma_{yz}) \\ & + n_z * (d_x * \sigma_{xz} + d_y * \sigma_{yz} + d_z * \sigma_{zz}) \end{aligned} \tag{2.4}$$

The resolved normal stress is

$$\begin{aligned} \sigma_{rns} = {} & n_x * (n_x * \sigma_{xx} + n_y * \sigma_{xy} + n_z * \sigma_{xz}) \\ & + n_y * (n_x * \sigma_{xy} + n_y * \sigma_{yy} + n_z * \sigma_{yz}) \\ & + n_z * (n_x * \sigma_{xz} + n_y * \sigma_{yz} + n_z * \sigma_{zz}) \end{aligned} \tag{2.5}$$

The resolved co-slip stress is

$$\begin{aligned}\tau_{rco} = n_x * (v_x * \sigma_{xx} + v_y * \sigma_{xy} + v_z * \sigma_{xz}) \\ + n_y * (v_x * \sigma_{xy} + v_y * \sigma_{yy} + v_z * \sigma_{yz}) \\ + n_z * (v_x * \sigma_{xz} + v_y * \sigma_{yz} + v_z * \sigma_{zz})\end{aligned} \tag{2.6}$$

Although not considered in this work, the components $\sigma_R(1,1)$, $\sigma_R(2,2)$, and $\sigma_R(1,2)$ are other resolved stress quantities.

2.4.4 Nucleation Stress

Once the slip system GB dislocation nucleation occurred (section 2.4.1) on and local stress is known for each simulation in both grains (section 2.4.2), the stress that causes GB dislocation nucleation to occur can be calculated. A nucleation time step is defined at the time the resolved shear stress on the nucleation slip system is maximal (for the simulations that nucleate on two slip systems simultaneously, the nucleation time step is taken as the earlier of the two maximal values. In practice, the maximal time steps were close to each other meaning either time step would work). The maximum shear correlates very well with the GB dislocation nucleation event. The nucleation stress (stress needed to activate GB dislocation nucleation) is taken as the stress just prior to this time step. The nucleation stress is not taken at the same time as the maximal shear because large stress fluctuations in the resolved shear and other stress components are expected to occur around the maximal shear time step (due to the GB accommodating stress via the GB dislocation nucleation mechanism). Taking the stress just prior allows us to avoid these expected fluctuations. The nucleation stress is defined separately for each grain.

The resolved shear stress (τ_{rss}), resolved normal stress (σ_{rns}) and resolved co-slip stress (τ_{rco}) at the nucleation time step for each of the twelve slip systems is stored for future analysis. These values constitute the resolved stress quantities that were present when GB dislocation nucleation occurs.

The final output of the process explained in this section is a database of the slip system of the GB dislocation nucleation event and the resolved stresses just prior to that event for every simulation and in each grain.

CHAPTER 3. RESULTS

3.1 Survey of Grain Boundary Dislocation Nucleation Responses

Representative GB dislocation nucleation responses for the 386 different triaxial stress states are presented in Figure 3.1. The right hand figures (3.1a, 3.1c, 3.1e) are snap shots of three different simulations at the point the stress was measured. The left hand figures (3.1b, 3.1d, 3.1f) are corresponding images after the nucleation event has progressed. Figure 3.1b shows a case where each grain nucleates partial dislocations on a single slip system. This accounts for about 50% of the simulations. Figure 3.1d shows a case where two different slip systems exhibit dislocation nucleation at the same time, or the two events occur so close together as to be nearly indistinguishable. This case accounts for about 45% or the simulations. Figure 3.1f is a special case of Figure 3.1d, where dislocation nucleation occurs on two different slip systems but the two slip systems share a common slip plane. In the structure that emerges from the GB in Figure 3.1f, the top slip plane has one Burgers vector and the bottom slip plane has another Burgers vector. Since the slip planes come in pairs, the middle plane acquires a full Burgers vector in between the two partial slip vectors. This occurred in about 40% of the simulations. Of this 40%, over 90% of the simulations could be categorized as initially starting on one of the two partials. The other simulations were analyzed in terms of nucleating on both slip systems. Finally, just under 4% of the simulations exhibited plasticity of some sort that was not readily categorizable onto one of the partial slip systems, these cases are excluded from the remaining analysis.

3.2 Nucleation System and Resolved Shear

Over the full range of 386 triaxial stress states, GB dislocation nucleation is observed on only six of the twelve possible slip systems in each grain. To show the full range of responses and the influence of the imposed stress states, the nucleation slip system for each stress state is

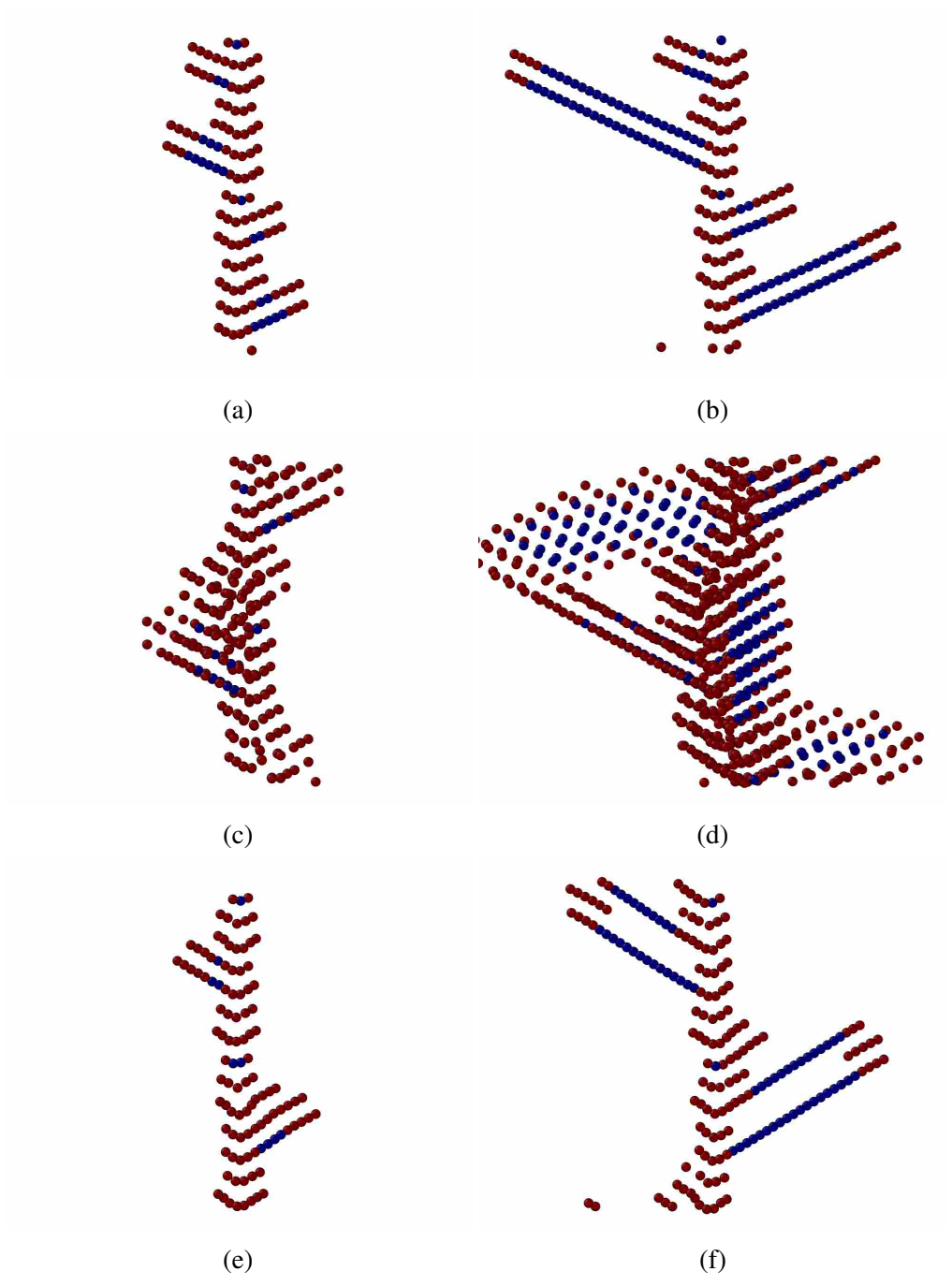


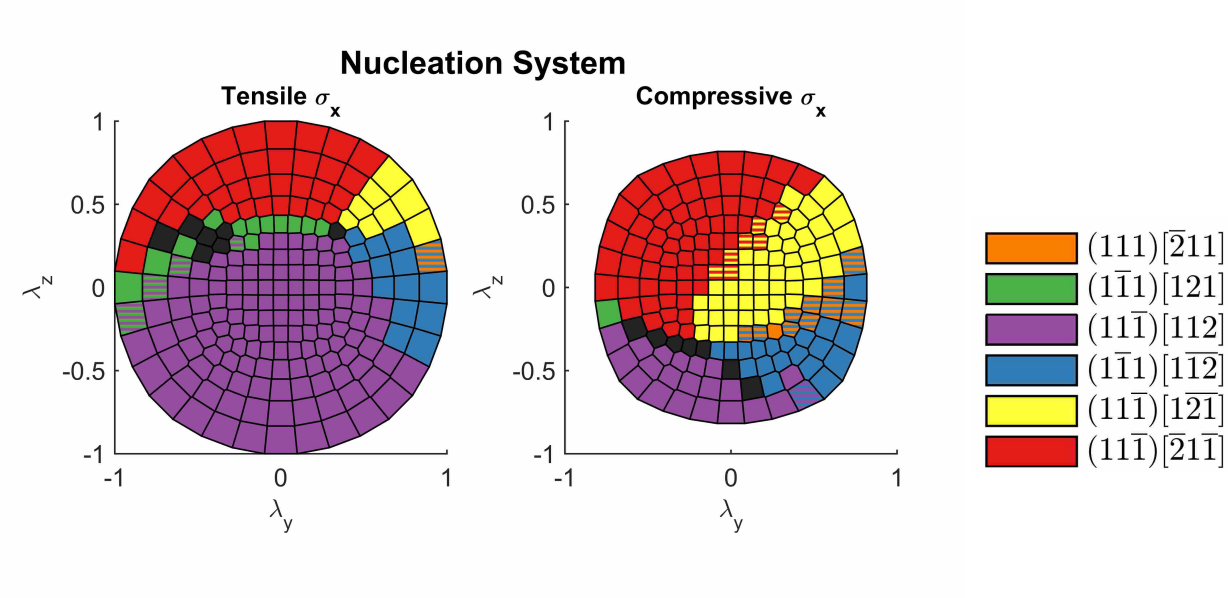
Figure 3.1: Some of the different nucleation events observed in simulations. The left column shows 3 simulations at the time the stress is measured. The right column shows the same simulations after the nucleation event has progressed

represented schematically in Figure 3.2. Since the target triaxial stress state is given by the λ vector, which resides on a unit sphere, the data is presented on stereographic maps for each grain (Figure 3.2a for grain A, and Figure 3.2b for grain B). The left map is for simulations with $\sigma_x \geq 0$, that is, tensile or zero σ_x. The right map is for simulations with $\sigma_x < 0$, that is, compressive σ_x. It is smaller than the left map because it omits the $\sigma_x = 0$ simulations, which are already present on the perimeter of the left map. The colors on the maps represent the different slip systems for GB dislocation nucleation, as given by the legend for each grain. The map has regions that are thatched with two colors for simulations that exhibited simultaneous nucleation on two slip systems. The black tiles represent simulations that are dropped from the analysis as mentioned in section 3.1.

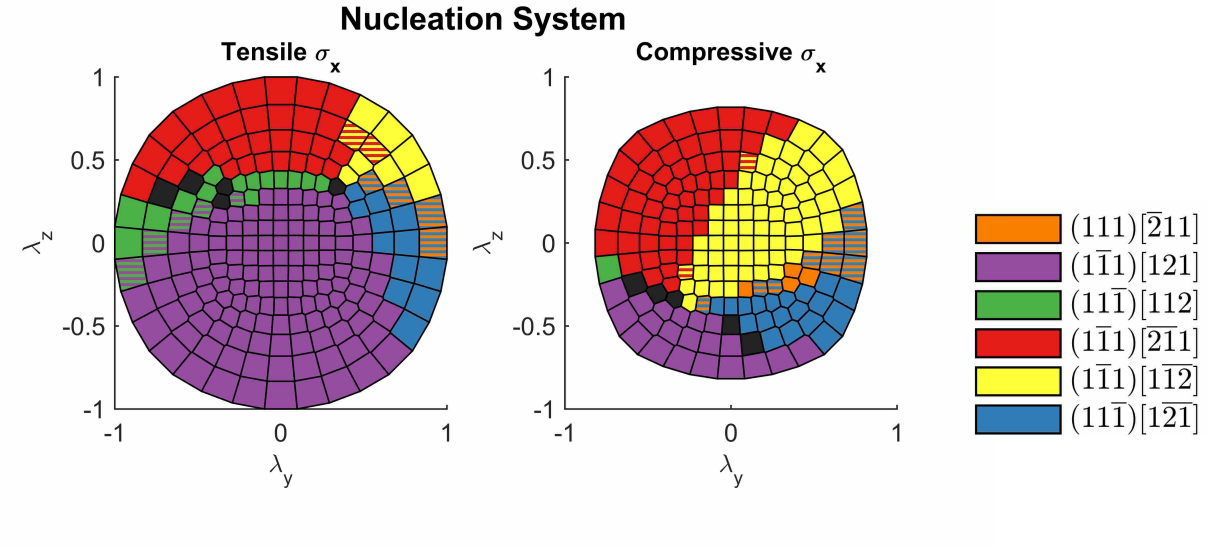
Comparing the two figures shows that there are complimentary slip systems likely due to the symmetry of the GB. For instance, if grain A nucleates on its red slip system, grain B will (with few exceptions) also nucleate on its red slip system (note that the red slip system in grain A is not the same slip system in grain B, colors have been chosen such that complimentary nucleation systems are the same color). Figures 3.3a and 3.3b show the magnitude of the resolved shear stress on the nucleation slip system just prior to nucleation. The variation of the resolved shear stress correlates well with the regions of same slip systems in Figures 3.2a and 3.2b. However, it is clear that different slip systems have different criteria for the required resolved shear stress. For instance, one slip system might require very high resolved shears to activate while another might require low shears. Also noteworthy is that even within a single nucleation system there is a gradient of resolved shear strengths needed for dislocation nucleation.

3.3 Resolved Stress Needed to Activate Grain Boundary Dislocation Nucleation

The variation of the resolved shear stresses, τ_{rss}, within a single slip system indicates that GB dislocation nucleation has non-Schmid behavior since a single critical resolved shear scalar does not predict nucleation. Other factors, such as the resolved normal, σ_{rns}, and co-slip, τ_{rco}, stresses are influencing the nucleation event as has been proposed and reported in literature [32,39]. Figure 3.4a plots τ_{rss}, σ_{rns} and τ_{rco} for grain A resolved on to the $(111)[\bar{2}11]$ slip systems for each analyzed triaxial stress state, regardless of whether the simulation actually nucleates on the $(111)[\bar{2}11]$ slip system. Those simulations that did nucleate on the $(111)[\bar{2}11]$ slip system are marked in closed circles, while those that did not are marked as open circles. When two slip



(a)

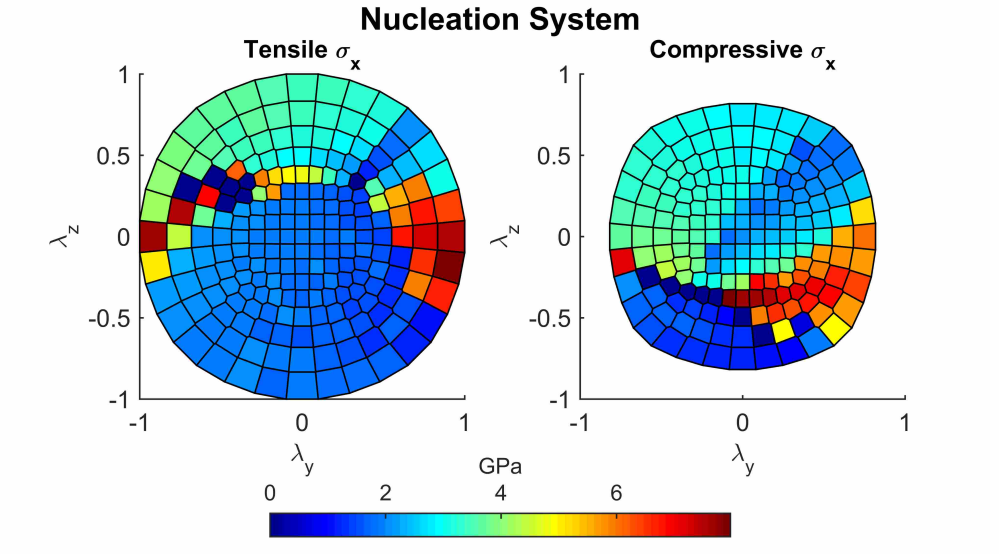


(b)

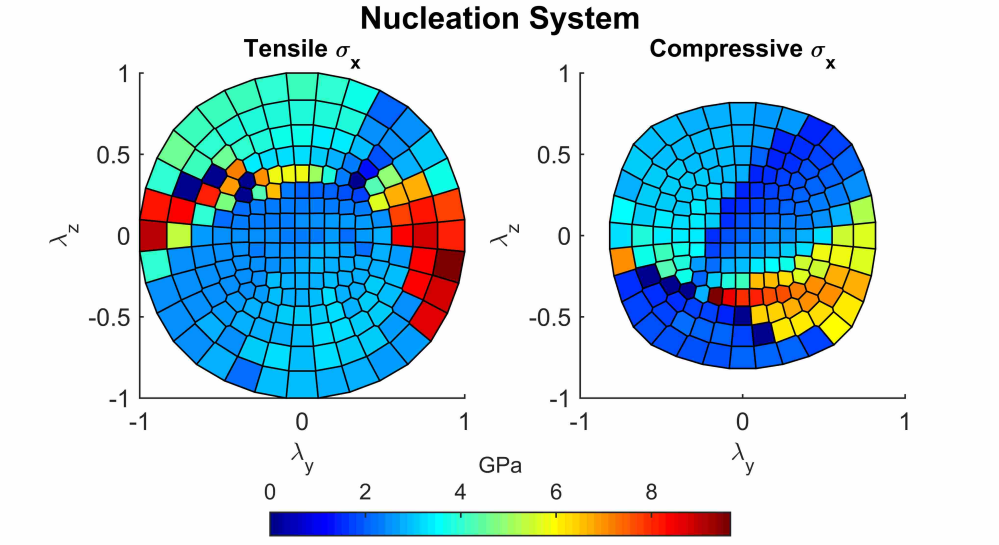
Figure 3.2: (a) The nucleation system for each triaxial stress state in grain A
(b) The nucleation system for each triaxial stress state in grain B

systems nucleate simultaneously, both slip systems are specified with a second ring for the other slip system. The slip system colors match those used in Figure 3.2a.

In examining Figure 3.4a further, it can be seen that the closed circles are almost always the highest points, and are clustered along what appears to be a plane. To demonstrate and quantify



(a)



(b)

Figure 3.3: (a) The resolved shear needed for nucleation for each triaxial stress state in grain A (b) The resolved shear needed for nucleation for each triaxial stress state in grain B

the normal and co-slip stress dependencies, a plane of the form $1 = \frac{\tau_{rss}}{\tau_{crss}} + \frac{\sigma_{rns}}{\sigma_{crns}} + \frac{\tau_{rco}}{\tau_{crco}}$ is fit to the closed circle points. The stresses of these closed circle points are the stresses needed to cause GB dislocation nucleation on the $(11\bar{1})[112]$ slip system. This is analogous to the way a critical

resolved shear stress is needed to activate slip in Schmid's law [51], except that the values needed for nucleation are predicted by a function instead of a single scalar quantity. The parameters τ_{crss}, σ_{crns}, and τ_{crco} are fitting parameters and correspond to the intercepts of the plane on the τ_{rss}, σ_{rns} and τ_{rco} axes, respectively.

This process is repeated for all slip systems on which nucleation occurs in both grains. Similar figures for the other 5 slip systems in grain A are presented in Figure 3.4b-f, while those for grain B are omitted for space. The planar fitting parameters for all nucleation slip systems in both grain A and grain B are shown in Table 3.1 and Table 3.2, respectively. These tables also contain the number of sample points, N, number of points above the plane that nucleate on some other slip system, N_{bad}, and the R^2 statistic for the planar fitting (note that this metric, while useful, should be interpreted with caution as it depends on the N value). Despite the visually appealing fit, the planar model does not always account for all variability (this is discussed further in section 4.3). Nonetheless, there are a small number of critical values that can describe the local stress dependence for nucleation on a given slip system.

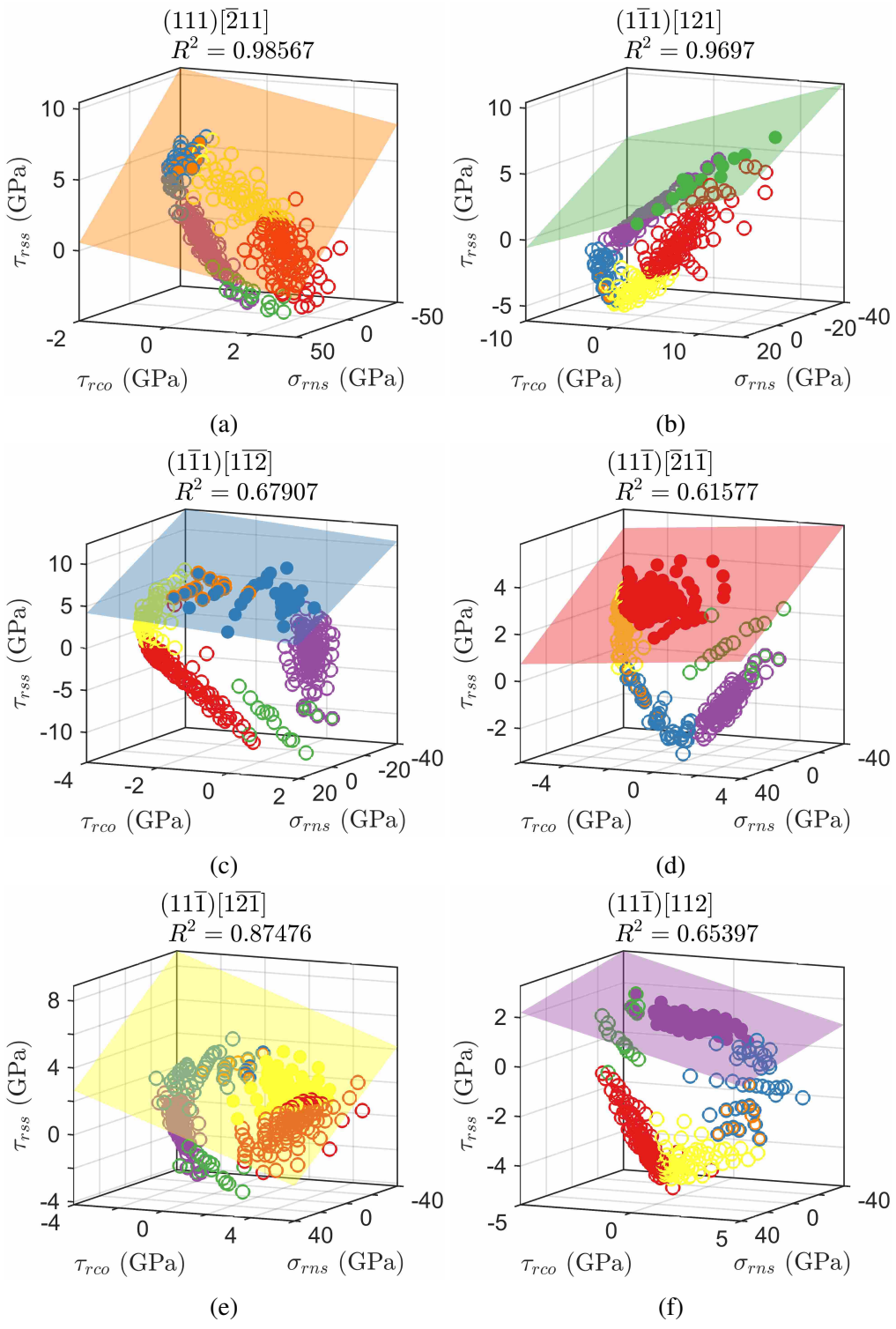


Figure 3.4: τ_{rss}, σ_{rns} and τ_{rco} for each nucleation slip system for GB dislocation nucleation into grain A. Points marked with a solid circle represent simulations that nucleated on the graph's slip system. A plane is fit to these points and represents the stress needed to activate grain boundary dislocation nucleation

Table 3.1: Coefficients found for each slip system's planar fit for GB dislocation nucleation into grain A. The coefficients quantify the normal and co-slip pressure dependence on the critical resolved shear stress

Slip System	τ_{crns}	σ_{crns}	τ_{crco}	N	N_{bad}	R^2 statistic
$(111)[\bar{2}11]$	4.38 GPa	44.3 GPa	7.7 GPa	12	8	0.986
$(1\bar{1}1)[121]$	3.42 GPa	35.6 GPa	-16.3 GPa	17	0	0.97
$(1\bar{1}1)[1\bar{1}\bar{2}]$	5.68 GPa	41.5 GPa	18.3 GPa	44	0	0.679
$(11\bar{1})[\bar{2}1\bar{1}]$	3.38 GPa	62.7 GPa	-42.3 GPa	106	0	0.616
$(11\bar{1})[1\bar{2}\bar{1}]$	3.88 GPa	49.7 GPa	8.19 GPa	62	1	0.875
$(11\bar{1})[112]$	1.58 GPa	117 GPa	6.79 GPa	154	2	0.654

Table 3.2: Coefficients found for each slip system's planar fit for GB dislocation nucleation into grain B

Slip System	τ_{crns}	σ_{crns}	τ_{crco}	N	N_{bad}	R^2 statistic
$(111)[\bar{2}11]$	4.57 GPa	44.8 GPa	-9.82 GPa	17	5	0.981
$(1\bar{1}1)[\bar{2}\bar{1}1]$	3.17 GPa	50.5 GPa	209 GPa	90	1	0.736
$(1\bar{1}1)[121]$	1.7 GPa	132 GPa	24.5 GPa	153	2	0.298
$(1\bar{1}1)[1\bar{1}\bar{2}]$	3.07 GPa	76.2 GPa	-11.1 GPa	74	12	0.482
$(11\bar{1})[1\bar{2}\bar{1}]$	5.93 GPa	42.1 GPa	-37.9 GPa	44	2	0.896
$(11\bar{1})[112]$	2.86 GPa	35.5 GPa	11.7 GPa	20	0	0.931

CHAPTER 4. DISCUSSION

4.1 Nucleation Criteria

The values of τ_{crss} represents the resolved shear stress needed for nucleation if both the resolved normal and co-slip stresses are zero. Likewise, σ_{crns} and τ_{crco} represent nucleation threshold values if the other two resolved quantities are zero. Since resolved shear stress is known to be the main factor in dislocation glide [52], it is unsurprising that the τ_{crss} fitting parameters are substantially smaller than the σ_{crns} and τ_{crcs} values. Initiating GB dislocation nucleation without any resolved shear stress is very hard. Intriguingly, the values of τ_{crss} vary dramatically from slip system to slip system. What is a large resolved shear stress for one slip system might be a small stress on another. This suggests that GB dislocation nucleation is favored on some slip systems over others.

As previously mentioned, Figure 3.4 shows that almost all simulations that nucleated on a slip system other than the graph they are plotted on (the asterisks) lie below the planar fit. The planar fit constitutes a nucleation criterion for a GB to nucleate a dislocation on the slip system the criterion is formulated for. As the stress in a simulation is ramped upwards, the magnitudes of τ_{rss}, σ_{rns}, and τ_{rco} grow on each slip system until their values satisfy one of the plane equations. The nucleation criterion is captured in the binary inequality

$$1 \leq \frac{\tau_{rss}}{\tau_{crss}} + \frac{\sigma_{rns}}{\sigma_{crns}} + \frac{\tau_{rco}}{\tau_{crco}} \tag{4.1}$$

When this inequality is true for any set of τ_{rss}, σ_{rns}, and τ_{rco}, GB dislocation nucleation is expected to occur on the slip system the inequality is formulated for.

With simple algebraic manipulation, equation (4.1) is changed into the functional form

$$\tau_{rss} = \tau_{crss} + \frac{\partial \tau_{crss}}{\partial \sigma_{crns}} \sigma_{rns} + \frac{\partial \tau_{crss}}{\partial \tau_{crco}} \tau_{rco} \tag{4.2}$$

where

$$\frac{\partial \tau_{crss}}{\partial \sigma_{crns}} = \frac{-\tau_{crss}}{\sigma_{crns}} \qquad \frac{\partial \tau_{crss}}{\partial \tau_{crco}} = \frac{-\tau_{crss}}{\tau_{crco}} \tag{4.3}$$

The coefficients $\frac{\partial \tau_{crss}}{\partial \sigma_{crns}}$ and $\frac{\partial \tau_{crss}}{\partial \tau_{crco}}$ explain how the resolved shear stress needed to nucleate a dislocation varies with resolved normal and co-slip stress. We have used a linear fitting equation so these partial derivatives are scalars. The value of $\frac{\partial \tau_{crss}}{\partial \sigma_{crns}}$ is expected to be negative (meaning larger compressive normal stresses raises the needed resolved shear stress) because compressive normal stresses should make it harder for the atoms on that slip system to glide past each other [32, 39]. Fitting parameters for each slip system are shown in Table 4.1 and Table 4.2. As expected, $\frac{\partial \tau_{crss}}{\partial \sigma_{crns}}$ is negative for each slip system. We do not attempt to theoretically motivate the sign of the $\frac{\partial \tau_{crss}}{\partial \sigma_{crco}}$ term because positive co-slip and negative co-slip should behave quite similarly.

Table 4.1: Functional fitting parameters for grain A

Slip System	τ_{crns}	$\frac{\partial \tau_{crss}}{\partial \sigma_{crns}}$	$\frac{\partial \tau_{crss}}{\partial \sigma_{crco}}$
$(111)[\bar{2}11]$	4.38 GPa	-0.0989	-0.569
$(1\bar{1}1)[121]$	3.42 GPa	-0.0962	0.21
$(1\bar{1}1)[1\bar{1}2]$	5.68 GPa	-0.137	-0.311
$(11\bar{1})[\bar{2}1\bar{1}]$	3.38 GPa	-0.054	0.08
$(11\bar{1})[1\bar{2}1]$	3.88 GPa	-0.078	-0.474
$(11\bar{1})[112]$	1.58 GPa	-0.0135	-0.233

Table 4.2: Functional fitting parameters for grain B

Slip System	τ_{crns}	$\frac{\partial \tau_{crss}}{\partial \sigma_{crns}}$	$\frac{\partial \tau_{crss}}{\partial \sigma_{crco}}$
$(111)[\bar{2}11]$	4.57 GPa	-0.102	0.466
$(1\bar{1}1)[\bar{2}1\bar{1}]$	3.17 GPa	-0.0628	-0.0151
$(1\bar{1}1)[121]$	1.7 GPa	-0.0128	-0.0695
$(1\bar{1}1)[1\bar{1}2]$	3.07 GPa	-0.0403	0.278
$(11\bar{1})[1\bar{2}1]$	5.93 GPa	-0.141	0.156
$(11\bar{1})[112]$	2.86 GPa	-0.0806	-0.245

4.2 Yield Surfaces

By combining the GB dislocation nucleation criteria developed in the Results chapter, a single yield criterion for GB dislocation nucleation is formed for the GB. If the yield criterion is known for a particular GB, it becomes trivial to know under what stress states and on what slip systems GB dislocation nucleation is expected to occur. A yield surface is built for the GB using principal stresses oriented in the $\hat{x}$, $\hat{y}$, and $\hat{z}$ directions. Note that the yield surface is not independent of orientation because of the presence of the GB. Yield surfaces for GB dislocation nucleation are shown in Figures 4.1 and 4.3 for both grains. Biaxial yield surface slices are included in Figures 4.2 and 4.4 to better show the structure of the yield surface. A line of hydrostatic stress is included for reference. The yield surface does not account for GB plasticity mechanisms other than the GB dislocation nucleation; however, if other plasticity mechanisms activation could be formulated as a function of stress state they could also be included.

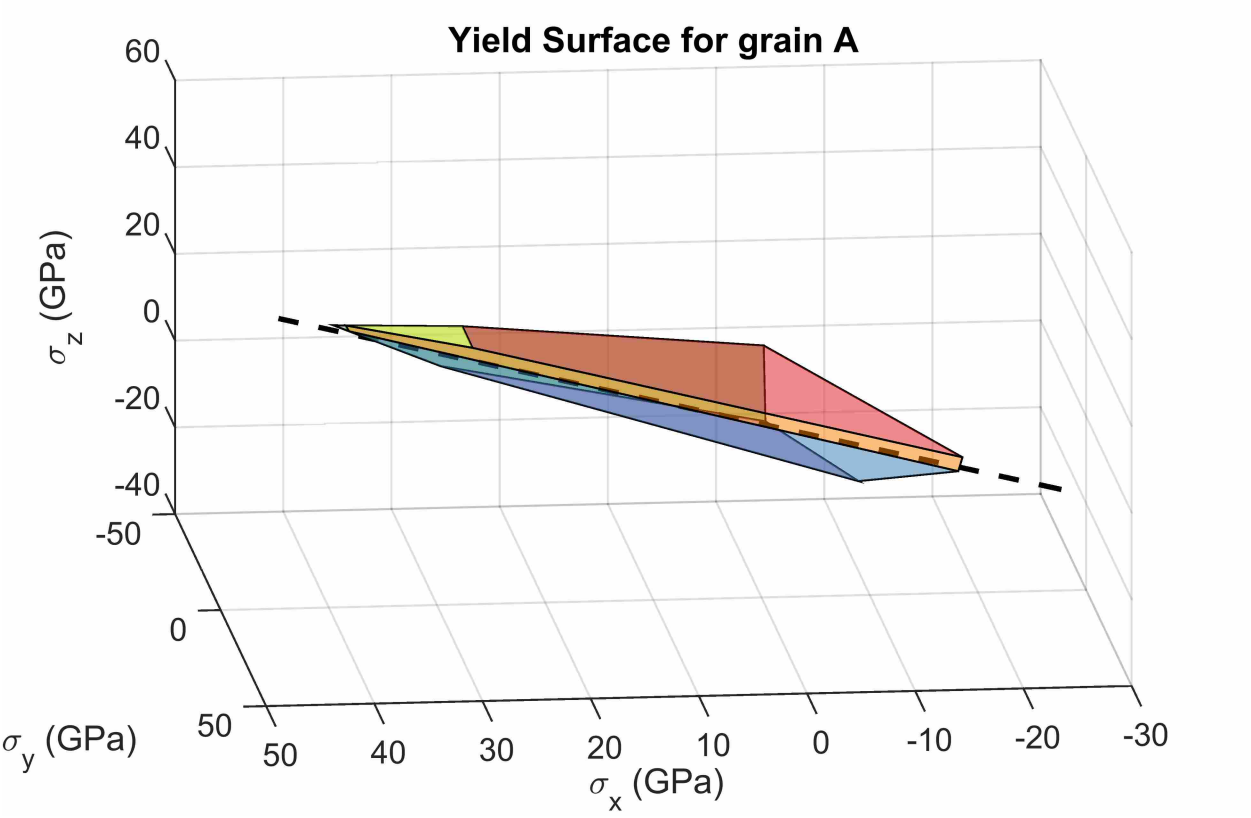


Figure 4.1: Theoretical yield surface for GB dislocation nucleation in grain A

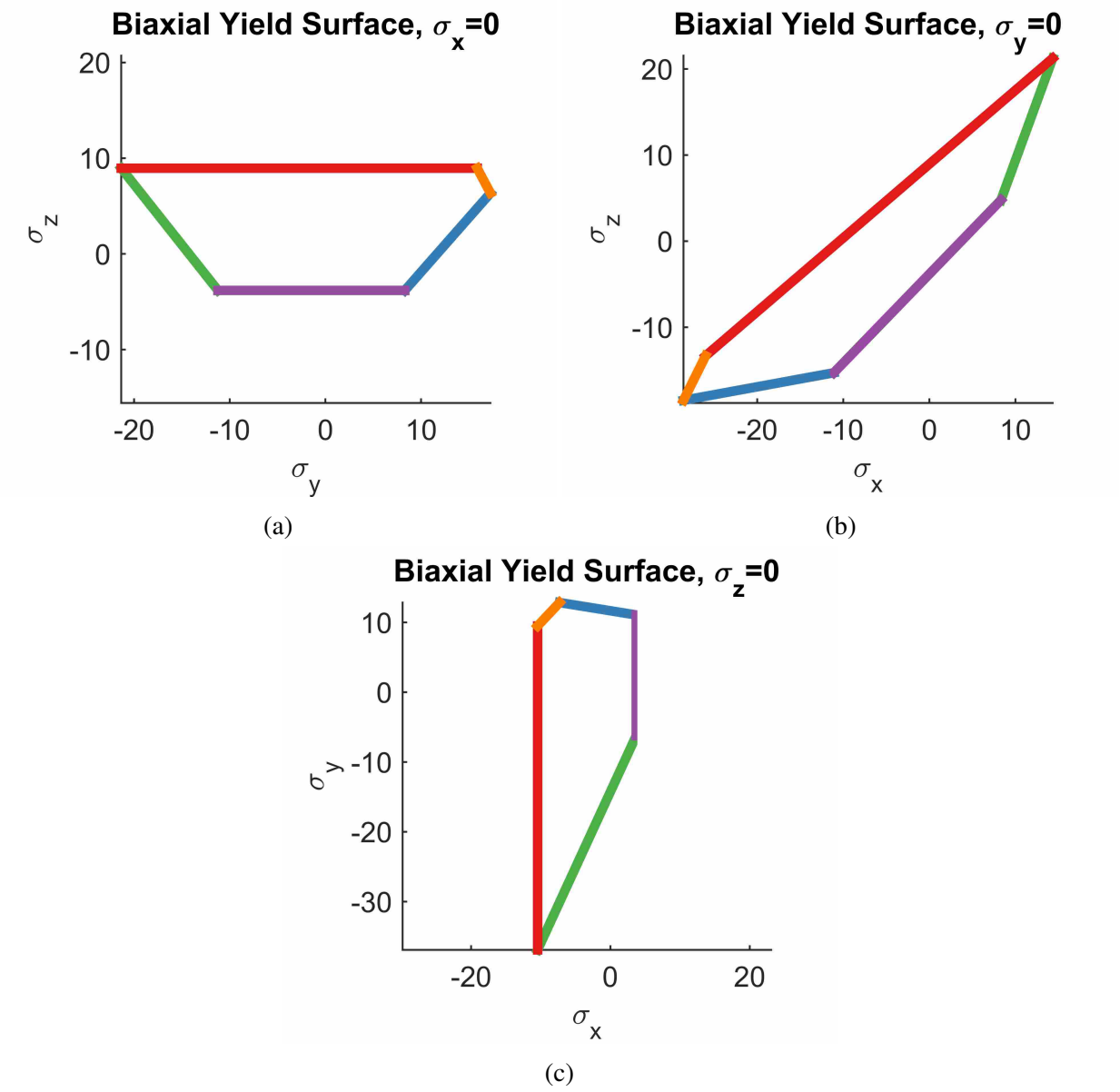


Figure 4.2: (a-c) Biaxial yield slices to illustrate the shape of the yield surface for grain A

The GB dislocation nucleation yield surface is reminiscent of a Mohr-Coulomb yield surface [53]. The yield surface is well oriented with the line of hydrostatic stress suggesting yield is largely a function of deviatoric stress, though not independent of other factors. Not surprisingly, the GB is particularly strong under triaxial compression; triaxial compression means there are large compressive resolved normal stresses which make it difficult for atoms to glide over each other. In Mohr-Coulomb, each face of the yield surface arises from changes in what two principal stresses control yield [53]; in our yield surface each face corresponds to GB dislocation nucleation on a

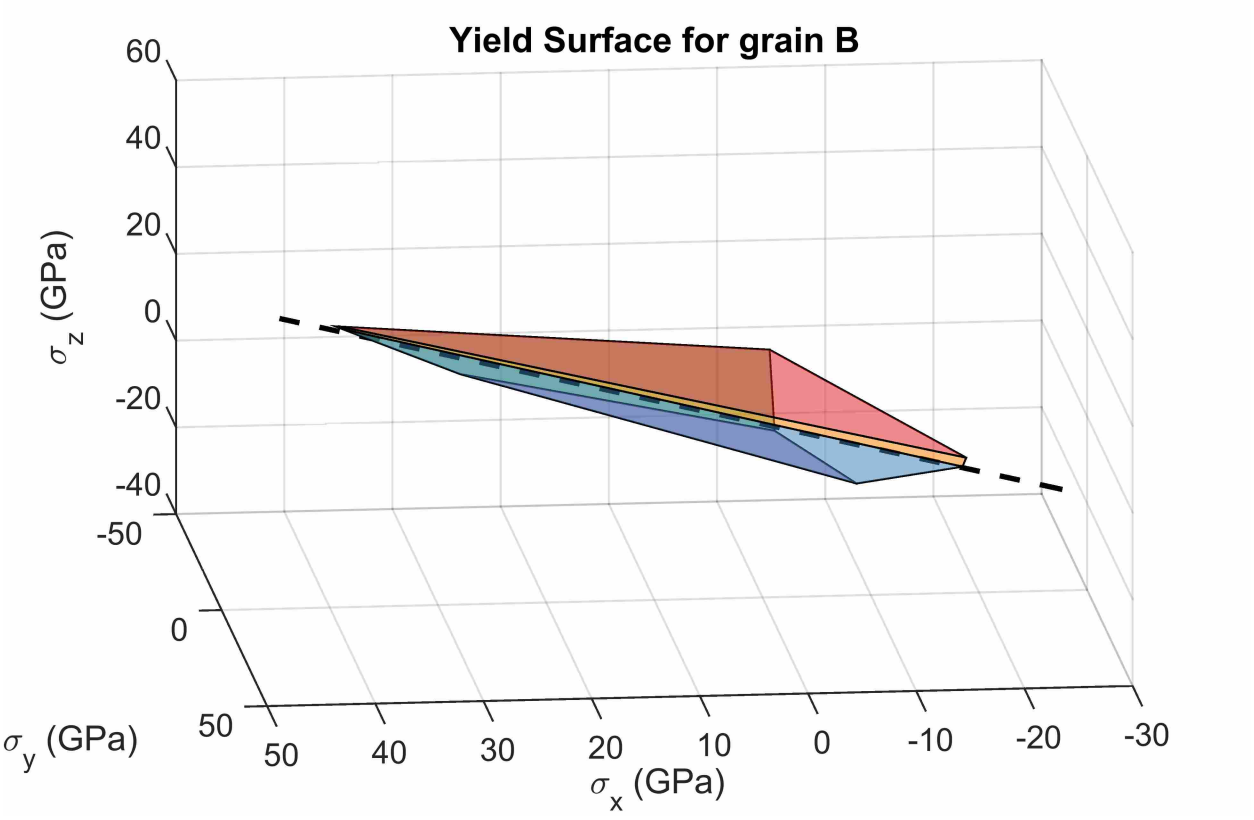


Figure 4.3: Theoretical yield surface for GB dislocation nucleation in grain B

different slip system. Note that while there where 6 nucleation criterion in grain A, there are only 5 faces on the yield surface. This is because in principal stress space one of the nucleation criterion was always activated before the unrepresented criterion was. The prior criterion essentially masked the latter. Unlike the Mohr-Coulomb yield surface, the GB dislocation nucleation yield surface does not end in a sharp point exactly on the line of hydrostatic stress. Instead, the tip is blunted with the individual faces not converging to a point. This is likely because each nucleation criterion has a different normal and co-slip stress dependence but may be due to noise in the data set.

4.3 Residual Analysis

The efficacy of using a planar GB dislocation nucleation criterion is analyzed using residual analysis. Residuals are defined as the error between a data point and a theoretical fit [54]. A model

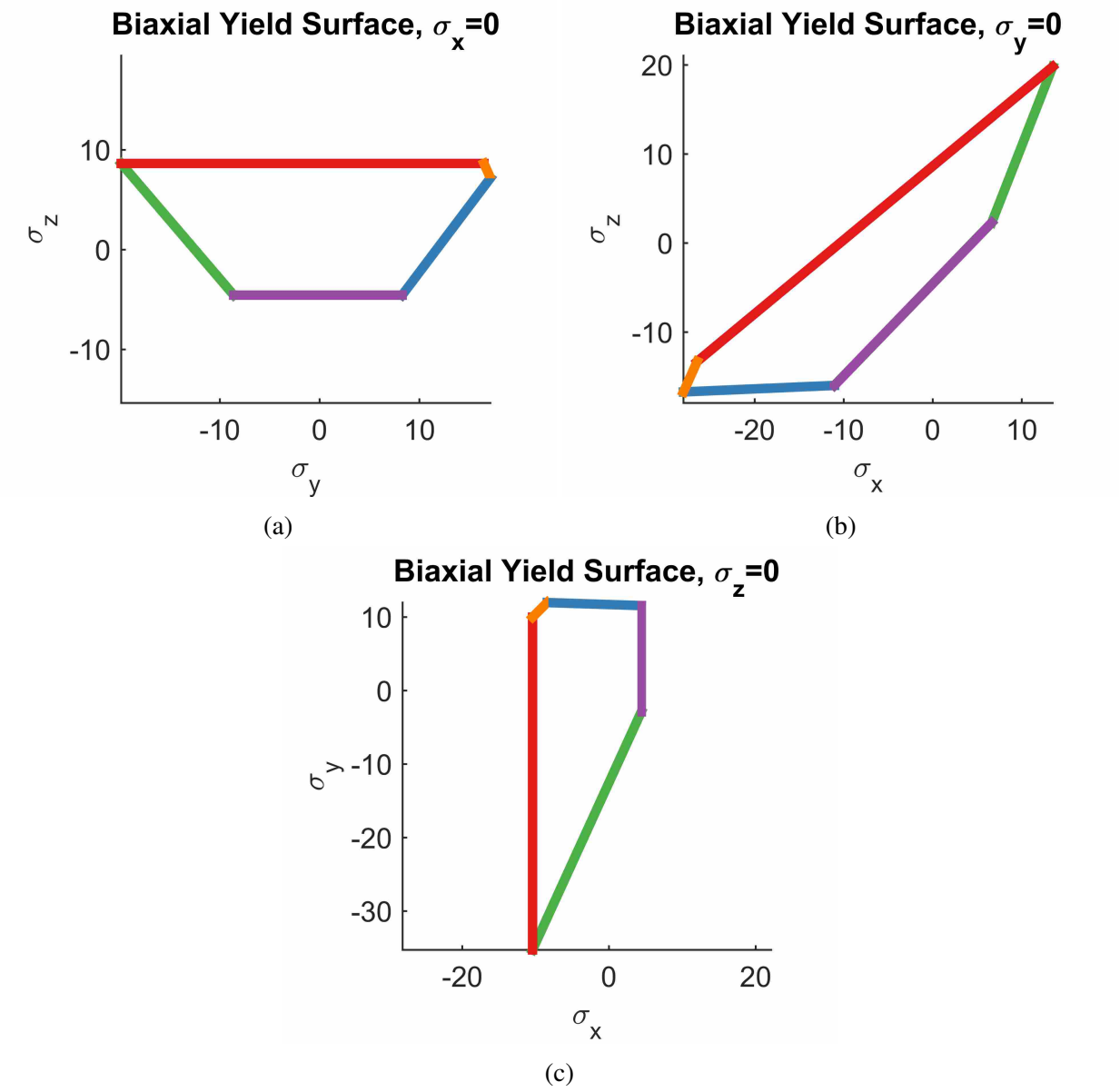


Figure 4.4: (a-c) Biaxial yield slices to illustrate the shape of the yield surface for grain B

is efficacious if its residuals are randomly dispersed. If there is a strong underlying pattern, then it means the model is failing to capture the shape of the data set.

A residual analysis is performed for each slip system. The residual error for the $(1\bar{1}1)[121]$ slip system is shown in Figure 4.5 with a polynomial surface of degree 2 fit to the data points. Note that the data points have a slight curve to them suggesting that the linear model used herein might be incorrect; however, looking at Figure 3.4 again, one will note that the curvature is very slight compared to the overall spread in the data. As such, we believe that using a planar nucleation

criterion model is acceptable. It is possible that a curved fit such as a paraboloid or ellipsoid would better model GB dislocation nucleation. This is left for future research.

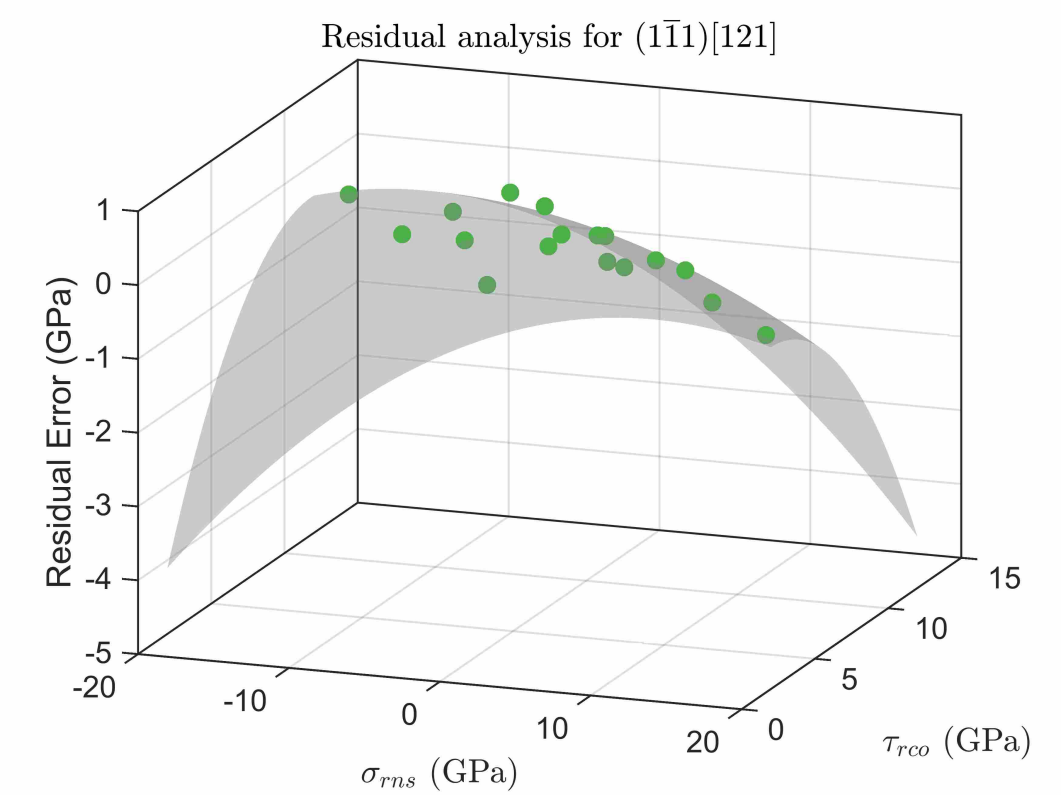


Figure 4.5: Residual error for the $(1\bar{1}1)[121]$ slip system. A polynomial surface of degree 2 is fit to the residual data

4.4 Stress Dependence and Grain Boundary Geometry

By plotting the values of τ_{crss} for each slip system as a function of the angle of the slip plane normal forms with the GB normal, we can attempt to find rules relating nucleation criteria (as captured by τ_{crss}) with the angular geometry of the GB. This is done for various angles in Figure 4.6 plotting τ_{crss}. The top right plot shows τ_{crss} as a function of the angle between an activated slip system's slip plane normal and GB normal (the $\hat{x}$ direction). The top left plot shows τ_{crss} as a function of the angle between slip plane direction and GB normal. The bottom two graphs do the same except the angle is formed with the GB's tilt axis (the $\hat{y}$ direction).

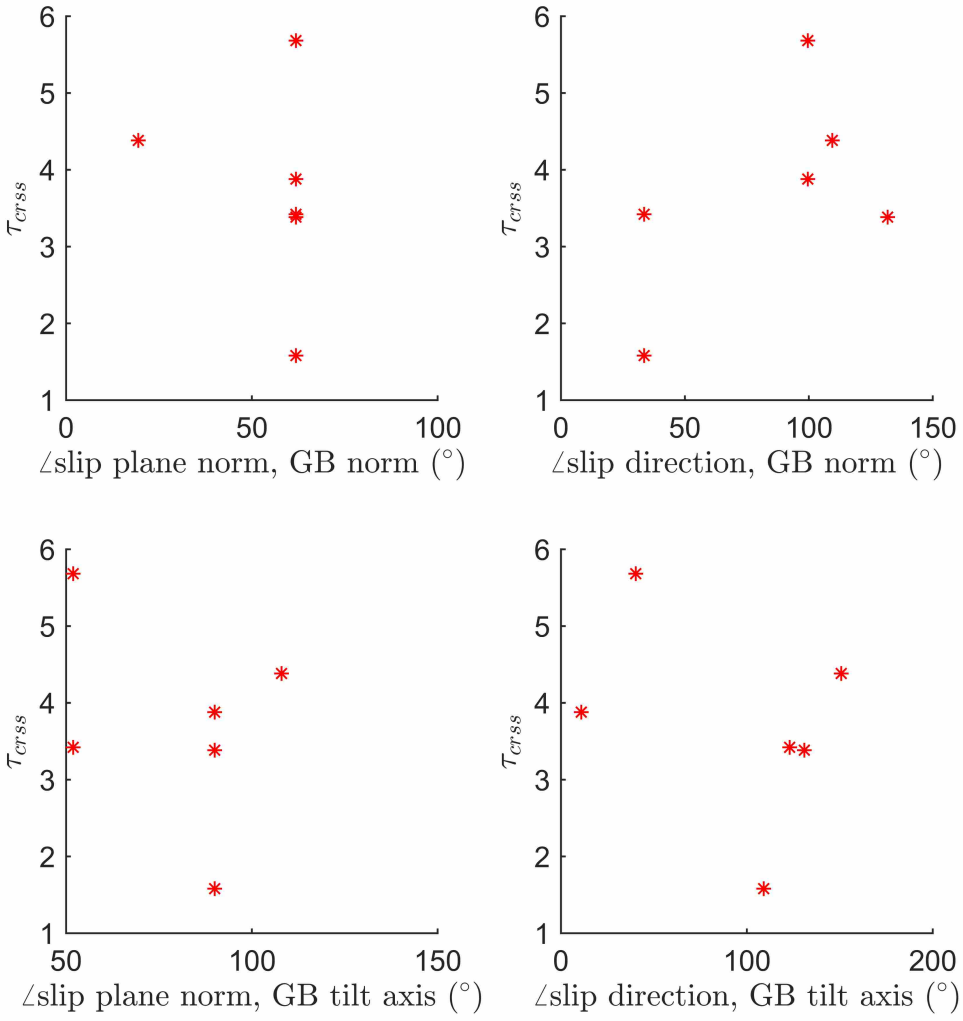


Figure 4.6: The shear intercept, τ_{crss} as a function of various angles the different nucleation slip systems form with the GB. There are no discernible patterns.

As is apparent, there appears to be no patterns in these graphs. This suggests that each slip system's τ_{crss} value cannot be found solely as a function of GB geometry (for example, the $(111)[1\bar{2}1]$ slip system's value of τ_{crss} cannot be geometrically correlated with the $(11\bar{1})[112]$ slip system's value of τ_{crss}); however, comparable slip systems from slightly different GB orientations (for example, the $(111)[1\bar{2}1]$ slip system's value of τ_{crss} for a 30° tilt GB can be correlated with the $(111)[1\bar{2}1]$ slip system's value of τ_{crss} for a 31° tilt GB) likely can be. Spearot, et. al. has already correlated maximum uniaxial stress (perpendicular to the GB) for GBs of differing tilt [32]. Graphs

are also made expressing $\frac{\partial \tau_{crss}}{\partial \sigma_{crns}}$ and $\frac{\partial \tau_{crss}}{\partial \tau_{crco}}$ as a function of angle with the GB yet are omitted for brevity. These graphs are similarly disordered.

4.5 Applicability to Other Systems and Phenomena

Besides GB dislocation nucleation, other GB plasticity mechanisms are known to exist (GB sliding, GB migration, etc.). Some of these mechanisms are known to activate under applied shear. Stress states in this paper are chosen to induce GB dislocation nucleation and suppress other plasticity mechanisms; specifically, stress states are chosen such that shear stresses are minimal. Incorporating shears into a unified GB plasticity yield criterion may be easy. For instance, GB sliding may have its own yield criterion, by Combining GB sliding's yield criterion with the GB dislocation nucleation criterion developed in this paper would likely produce a yield criterion that predicts GB sliding as well as GB dislocation nucleation. An exhaustive, five dimensional survey of the possible ways to ramp normal and shear stresses on a GB is needed.

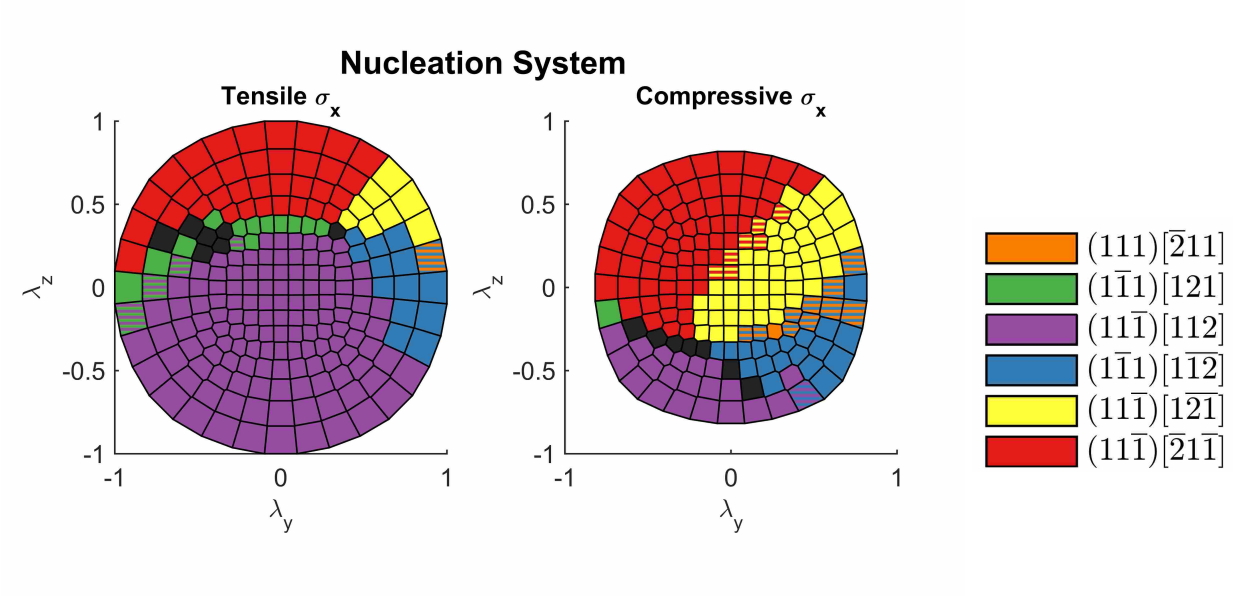
We do not know if this work will transfer to more complex systems containing high energy GB structures including non-equilibrium interfaces [42], GB ledges [3, 55], and triple junctions [34]. Of necessity, this paper has taken the limited scope of analyzing a single, idealized planar GB under the large but not exhaustive set of possible triaxial stress states. It is quite possible that non-equilibrium systems would conform to the planar GB dislocation nucleation criterion formulated herein and would simply have different values for τ_{crss}, σ_{crns}, and τ_{crco}. It is also possible that the high energy structures nucleate based on a different set of relevant physics that would make this work inapplicable to such systems. In one example, Tucker, et. al. showed that by adding non-equilibrium content to a GB, its primary nucleation mechanism changed from GB dislocation nucleation to GB migration [35]. We have no reason to believe that these findings should not generalize to GB dislocation nucleation from most other planar equilibrium GBs. Exceptions could include GBs with abnormal structures such as extremely high nanoporosity [40], GBs known to shear couple [56], or GBs with dissociated structure [49, 50].

4.6 Nucleation System and Schmid Factor

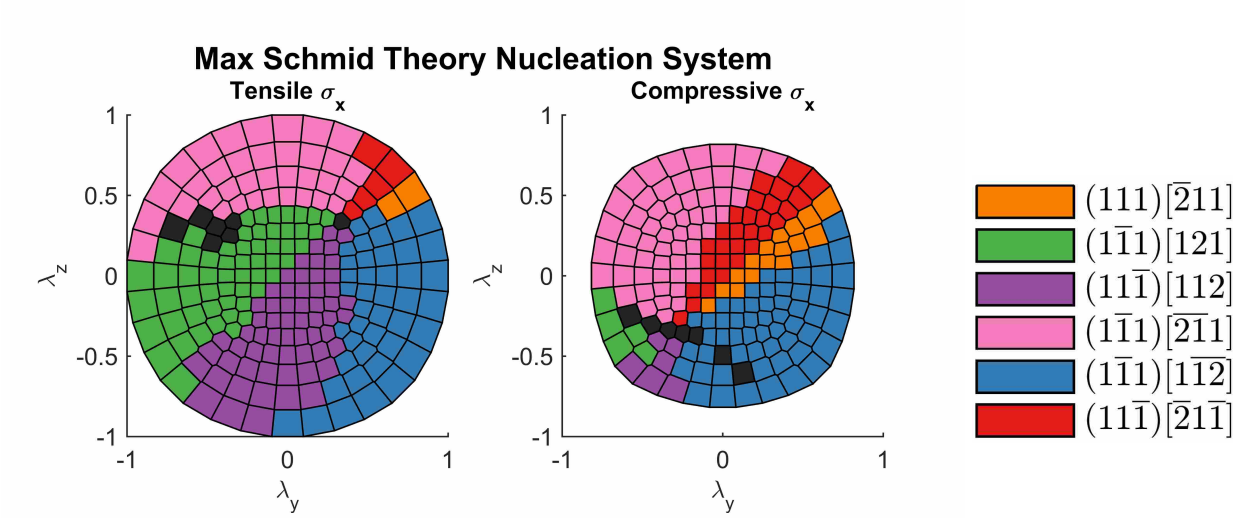
The Schmid factor has been proposed as a useful metric in determining what slip system GB dislocation nucleation occurs on [42, 57]. The metric is clearly important in uniaxial tension; however, many have found cases where nucleation occurs on a slip system other than would be predicted by a Schmid factor analysis [33, 41, 50]. The yield criterion developed herein shows that Schmid factor does not predict nucleation system as the activated GB dislocation nucleation system is also a function of normal and co-slip stresses. Figure 4.7a shows the slip system that is actually nucleated on in grain A. Figure 4.7b shows the expected slip system based on a maximal Schmid analysis using the local stress for grain A. Some regions of the stereographic plot show that the system predicted by maximal Schmid match the actual nucleation system but the regions for each slip system are generally too small or too large. More importantly, the maximum Schmid treatment suggests nucleation should occur on slip systems that are never actually nucleated on in the simulations. We believe that maximum Schmid theories to predict nucleation system are inaccurate.

4.7 Effect of Stress Definition

In the 2.4.2 section, we used a local stress defined only using atoms close to the GB to get the stress the GB was under. Due to the GB back stress, the stress at the GB compared to the pressure the bicrystal is under can vary considerably. Here, we repeat the analysis using different stress definitions to see what effect the chosen stress definition has on the nucleation criteria. We calculate τ_{crss}, σ_{crns}, and τ_{crco} for each slip system using virial stresses averaged over a 5-10 Å band (the same used originally, which corresponds to a standoff of 7.5 Å), a 15-20 Å band (standoff of 17.5 Å), a 25-30 Å band (standoff of 27.5 Å), and a 35-40 Å band (standoff of 37.5 Å) where the time nucleation occurs is tied to the maximal shear stress on the nucleation system. A global stress using the pressure the bicrystal is under is also included; however, the time of nucleation is kept at the same time as the 5-10 Å band. This was done because when using the global stress, the expected drop in resolved shear stress at the point of nucleation is very small because the nucleation event is averaged out over every atom in the simulation.



(a)



(b)

Figure 4.7: (a) the actual nucleation system that activated in grain A (b) the theoretically predicted slip system according to a max Schmid treatment. Note that the graphs do not match up very well showing that maximum Schmid theories to predict nucleation system are inaccurate

Comparisons of these different stress definitions are shown in Figure 4.8 for grain A. Note that the 15-20 Å band, 25-35 Å band, and 35-40 Å band predict similar fitting parameters to the original 5-10 Å band. This suggests that the methodology developed herein is relatively insensitive to local stress definition; however, the global pressure only matches the others some times. For

instance, the global fitting parameters for the $(1\bar{1}1)[121]$ slip system (green) consistently match the fitting parameters for the local stress definitions as seen in the green graphs; but the global fitting parameters for the $(11\bar{1})[\bar{2}1\bar{1}]$ slip system (red) vary substantially from the other stress definitions as seen by the global bar not matching the other bars in the red graphs. It appears that a global stress definition may work for GB dislocation nucleation on some slip systems and not for others. We recommend using a local stress taken from within the grain dislocations are being nucleated into.

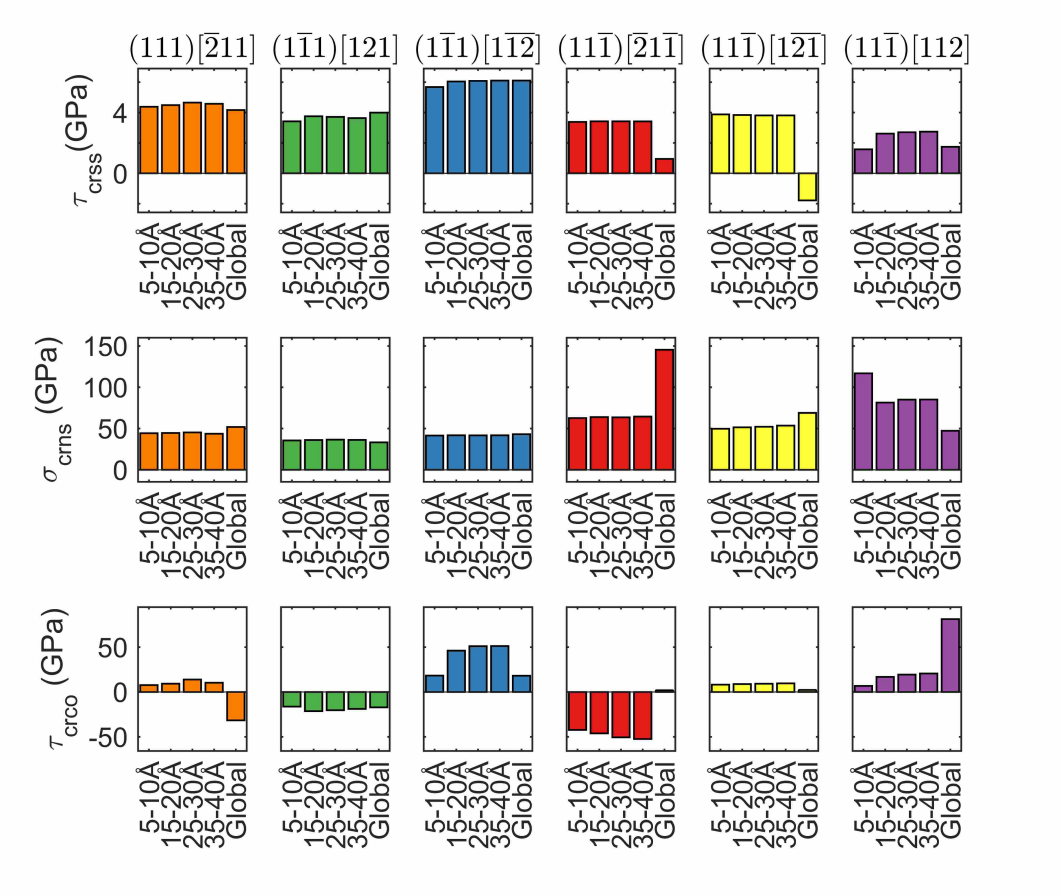


Figure 4.8: Comparison of different stress definitions on the fitting parameters τ_{crss}, σ_{crns}, and τ_{crss}. Each row of graphs is for a different fitting parameter. Each column of graphs is for a different slip system. Bars are for different stress definitions as defined in the text

CHAPTER 5. CONCLUSION

5.1 Summary of Work

Molecular dynamics is used to measure grain boundary dislocation nucleation under continuously variable triaxial stresses. This allows us to quantify the resolved shear, normal, and co-slip stresses needed for a GB to nucleate dislocations. A summary of this thesis is as follows:

- Applying triaxial stresses to a bicrystal activates the GB dislocation nucleation plasticity mechanism
- The slip system GB dislocation nucleation occurs on can be found by analyzing the stacking fault left behind a nucleated partial dislocation
- The threshold stress that leads to GB dislocation nucleation is taken as the stress when resolved shear on the nucleation system is maximal. From this, the resolved stresses that lead to GB dislocation nucleation are calculated
- We use a linear function of resolved shear, normal, and co-slip stresses to predict GB dislocation nucleation
- Criteria for each nucleation system are formulated. Each criteria is found to have different fitting parameters
- An algebraic manipulation of the nucleation criterion allows us to verify the fitting parameters are consistent with theory
- A residual analysis suggests the linear form of the nucleation criteria fits reasonably well; however, better non-linear forms may be possible
- By combining criteria for each slip system together, a theoretical yield surface for GB dislocation nucleation can be generated

- Finding fitting parameters for different local stress definitions as well as a global stress definition shows that the fitting parameters should be taken using a local stress within the grain dislocations are nucleated into

The theoretical yield surface is reminiscent of the Mohr-Coulomb yield surface. Each face of the surface corresponds to GB dislocation nucleation on a different slip system. By analyzing simulated slip system and comparing with the slip system that would be predicted by Schmid Factor, it is shown that Schmid factor poorly predicts nucleation system.

5.2 Future Work

The criteria developed herein will allow others to predict GB dislocation nucleation without having to resort to expensive computational simulations. By formulating GB dislocation nucleation yield constitutive laws, we hope that finite element crystal plasticity models [58] might be able to incorporate GB dislocation nucleation effects. Before this can be done, nucleation criteria for many different GBs must be determined.

We have demonstrated that applying many different triaxial stress states to a GB allows us to build criteria for GB dislocation nucleation. Beyond GB dislocation nucleation, it is attractive to formulate yield criteria for other types of GB plasticity mechanisms such as GB sliding and GB migration. This work specifically attempted to suppress such mechanisms; however, these mechanisms could be studied by applying shears and normal stresses to a bicrystal system.

5.3 Major Findings

To our knowledge, this work constitutes the first time that criteria for GB dislocation nucleation activation have been formulated under triaxial stress. A nucleation criterion exists for each slip system the GB might nucleate on. Each criterion has a different set of fitting parameters. Use of the criteria allows prediction of what stresses and on what slip systems GB dislocation nucleation is expected to occur on. Using the nucleation criteria, we build theoretical yield surfaces for GB dislocation nucleation.

We have definitively demonstrated that GB dislocation nucleation depends on resolved shear, resolved normal, and resolved co-slip stresses further validating work done by Spearot, et.

al. [32]. We show that the slip system GB dislocation nucleation occurs on cannot be determined by maximum Schmid factor alone.

5.4 Funding

This work was supported by the U.S. Department of Energy, Office of Science, Basic Energy Sciences (http://science.energy.gov/bes/) under Award #DE-SC0012587.

REFERENCES

[1] Kroner, E., and Anthony, K. H., 1975. "Dislocations and Disclinations in Material Structures: The Basic Topological Concepts." *Annual Review of Materials Science,* **5**(1), pp. 43–72. 1

[2] Callister, W., and Rethwisch, D., 2009. *Materials Science and Engineering: An Introduction, 8th Edition*. Wiley. 1

[3] Hunter, A., and Beyerlein, I., 2014. "Stacking fault emission from grain boundaries: Material dependencies and grain size effects." *Materials Science and Engineering: A,* **600**, apr, pp. 200–210. 1, 32

[4] Hoagland, R. G., and Valone, S. M., 2015. "Emission of dislocations from grain boundaries by grain boundary dissociation." *Philosophical Magazine,* **95**(February), pp. 112–131. 1

[5] Tschopp, M., and McDowell, D., 2008. "Grain boundary dislocation sources in nanocrystalline copper." *Scripta Materialia,* **58**(4), feb, pp. 299–302. 1

[6] Partial dislocations and stacking faults http://www.tf.uni-kiel.de/matwis/amat/def_en/kap_5/backbone/r5_4_1.html Accessed: 2016-06-01. 1

[7] Capolungo, L., Spearot, D., Cherkaoui, M., McDowell, D., Qu, J., and Jacob, K., 2007. "Dislocation nucleation from bicrystal interfaces and grain boundary ledges: Relationship to nanocrystalline deformation." *Journal of the Mechanics and Physics of Solids,* **55**(11), nov, pp. 2300–2327. 2, 5, 11

[8] Rohrer, G. S., 2011. "Grain boundary energy anisotropy: a review." *Journal of Materials Science,* **46**(18), pp. 5881–5895. 2

[9] Olmsted, D. L., Foiles, S. M., and Holm, E. A., 2009. "Survey of computed grain boundary properties in face-centered cubic metals: I. Grain boundary energy." *Acta Materialia,* **57**(13), aug, pp. 3694–3703. 3, 7

[10] Olmsted, D. L., Holm, E. A., and Foiles, S. M., 2009. "Survey of computed grain boundary properties in face-centered cubic metals - II: Grain boundary mobility." *Acta Materialia,* **57**(13), aug, pp. 3704–3713. 3, 7

[11] Swope, W. C., Andersen, H. C., Berens, P. H., and Wilson, K. R., 1982. "A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters." *The Journal of Chemical Physics,* **76**(1), pp. 637–649. 3

[12] Tadmor, E. B., and Miller, R. E., 2011. *Modeling Materials*. Cambridge University Press Cambridge Books Online. 3, 4

[13] Jones, J. E., 1924. "On the determination of molecular fields. ii. from the equation of state of a gas." *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences,* **106**(738), pp. 463–477. 3

[14] van Duin, A. C. T., Dasgupta, S., Lorant, F., and Goddard, W. A., 2001. "Reaxff: A reactive force field for hydrocarbons." *The Journal of Physical Chemistry A,* **105**(41), pp. 9396–9409. 3

[15] Becker, C. A., Tavazza, F., Trautt, Z. T., and Buarque de Macedo, R. A., 2013. "Considerations for choosing and using force fields and interatomic potentials in materials science and engineering." *Current Opinion in Solid State and Materials Science,* **17**(6), dec, pp. 277–283. 3

[16] Foiles, S., and Hoyt, J., 2006. "Computation of grain boundary stiffness and mobility from boundary fluctuations." *Acta Materialia,* **54**(12), jul, pp. 3351–3357. 3, 7

[17] Van Swygenhoven, H., Derlet, P. M., and Frøseth, A. G., 2004. "Stacking fault energies and slip in nanocrystalline metals.." *Nature materials,* **3**(6), pp. 399–403. 3

[18] Sangid, M. D., Ezaz, T., Sehitoglu, H., and Robertson, I. M., 2011. "Energy of slip transmission and nucleation at grain boundaries." *Acta Materialia,* **59**(1), jan, pp. 283–296. 3, 5, 6

[19] Daw, M. S., and Baskes, M. I., 1984. "Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals." *Phys. Rev. B,* **29**, Jun, pp. 6443–6453. 4

[20] Daw, M. S., and Baskes, M. I., 1984. "Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals." *Phys. Rev. B,* **29**, Jun, pp. 6443–6453. 4

[21] Plimpton, S., 1995. "Fast Parallel Algorithms for Short-Range Molecular Dynamics." *Journal of Computational Physics,* **117**(1), mar, pp. 1–19. 4, 7

[22] Zimmerman, J. A., Kelchner, C. L., Klein, P. A., Hamilton, J. C., and Foiles, S. M., 2001. "Surface step effects on nanoindentation." *Physical review letters,* **87**(16). 4, 10, 51, 63

[23] Brown, W. M., Wang, P., Plimpton, S. J., and Tharrington, A. N., 2011. "Implementing molecular dynamics on hybrid high performance computers - short range forces." *Computer Physics Communications,* **182**(4), apr, pp. 898–911. 4, 7

[24] Brown, W. M., Kohlmeyer, A., Plimpton, S. J., and Tharrington, A. N., 2012. "Implementing molecular dynamics on hybrid high performance computers - Particle-particle particle-mesh." *Computer Physics Communications,* **183**(3), mar, pp. 449–459. 4, 7

[25] Brown, W. M., and Yamada, M., 2013. "Implementing molecular dynamics on hybrid high performance computers - Three-body potentials." *Computer Physics Communications,* **184**(12), dec, pp. 2785–2793. 4, 7

[26] Fulton supercomputing lab https://marylou.byu.edu/ Accessed: 2016-06-01. 4

[27] Slurm workload manager http://slurm.schedmd.com/ Accessed: 2016-06-01. 4

[28] Voyiadjis, G. Z., and Yaghoobi, M., 2016. "Role of grain boundary on the sources of size effects." *Computational Materials Science,* **117**, may, pp. 315–329. 5

[29] Van Swygenhoven, H., 2002. "Grain Boundaries and Dislocations." *Science,* **296**(5565), pp. 66–67. 5

[30] Lund, A., and Schuh, C., 2005. "Strength asymmetry in nanocrystalline metals under multiaxial loading." *Acta Materialia,* **53**(11), jun, pp. 3193–3205. 5

[31] Wolf, D., Yamakov, V., Phillpot, S., Mukherjee, A., and Gleiter, H., 2005. "Deformation of nanocrystalline materials by molecular-dynamics simulation: relationship to experiments." *Acta Materialia,* **53**(1), jan, pp. 1–40. 5

[32] Spearot, D., Tschopp, M., Jacob, K., and McDowell, D., 2007. "Tensile strength of $< 100 >$ and $< 110 >$ tilt bicrystal copper interfaces." *Acta Materialia,* **55**(2), jan, pp. 705–714. 5, 18, 25, 31, 38

[33] Beyerlein, I. J., Wang, J., and Zhang, R., 2013. "Mapping dislocation nucleation behavior from bimetal interfaces." *Acta Materialia,* **61**(19), nov, pp. 7488–7499. 5, 6, 33

[34] Wu, Z., Zhang, Y., Jhon, M., and Srolovitz, D., 2013. "Anatomy of nanomaterial deformation: Grain boundary sliding, plasticity and cavitation in nanocrystalline Ni." *Acta Materialia,* **61**(15), sep, pp. 5807–5820. 5, 6, 32

[35] Tucker, G. J., and McDowell, D. L., 2011. "Non-equilibrium grain boundary structure and inelastic deformation using atomistic simulations." *International Journal of Plasticity,* **27**(6), jun, pp. 841–857. 5, 32

[36] Tschopp, M. A., Spearot, D. E., and McDowell, D. L., 2007. "Atomistic simulations of homogeneous dislocation nucleation in single crystal copper." *Modelling and Simulation in Materials Science and Engineering*, pp. 693–709. 5

[37] Tschopp, M., and McDowell, D., 2008. "Influence of single crystal orientation on homogeneous dislocation nucleation under uniaxial loading." *Journal of the Mechanics and Physics of Solids,* **56**(5), may, pp. 1806–1830. 5

[38] Ogata, S., Li, J., and Yip, S., 2002. "Ideal pure shear strength of aluminum and copper." *Science (New York, N.Y.),* **298**(5594), pp. 807–811. 5

[39] Kinoshita, K., Shimokawa, T., and Kinari, T., 2012. "Grain Boundary Structure Dependence of Extrinsic Grain Boundary Dislocation Emission Phenomena: A Molecular Dynamics Study." *Materials Transactions,* **53**(1), pp. 147–155. 5, 7, 18, 25

[40] Spearot, D. E., 2008. "Evolution of the E structural unit during uniaxial and constrained tensile deformation." *Mechanics Research Communications,* **35**(1-2), jan, pp. 81–88. 5, 32

[41] Zhang, R., Wang, J., Beyerlein, I., and Germann, T., 2011. "Dislocation nucleation mechanisms from fcc/bcc incoherent interfaces." *Scripta Materialia,* **65**(11), dec, pp. 1022–1025. 6, 33

[42] Burbery, N., Das, R., and Ferguson, W., 2015. "Modelling with variable atomic structure: Dislocation nucleation from symmetric tilt grain boundaries in aluminium." *Computational Materials Science,* **101**, apr, pp. 16–28. 6, 32, 33

[43] Faken, D., and Jónsson, H., 1994. "Systematic analysis of local atomic structure combined with 3D computer graphics." *Computational Materials Science,* **2**(2), mar, pp. 279–286. 7, 10

[44] Stukowski, A., 2009. "Visualization and analysis of atomistic simulation data with OVITO-the Open Visualization Tool." *Modelling and Simulation in Materials Science and Engineering,* **18**(1), p. 015012. 7

[45] Semechko, A., 2015. Suite of functions to perform uniform sampling of a sphere https://www.mathworks.com/matlabcentral/fileexchange/37004-suite-of-functions-to-perform-uniform-sampling-of-a-sphere Accessed: 2016-04-19. 8, 44

[46] Zheng, G. P., Wang, Y. M., and Li, M., 2005. "Atomistic simulation studies on deformation mechanism of nanocrystalline cobalt." *Acta Materialia,* **53**(14), pp. 3893–3901. 10

[47] Van Swygenhoven, H., Derlet, P., and Frøseth, A., 2006. "Nucleation and propagation of dislocations in nanocrystalline fcc metals." *Acta Materialia,* **54**(7), apr, pp. 1975–1983. 10

[48] Thompson, A. P., Plimpton, S. J., and Mattson, W., 2009. "General formulation of pressure and stress tensor for arbitrary many-body interaction potentials under periodic boundary conditions." *The Journal of chemical physics,* **131**(15), oct, p. 154107. 12

[49] Tschopp, M., and McDowell, D., 2008. "Dislocation nucleation in Σ3 asymmetric tilt grain boundaries." *International Journal of Plasticity,* **24**(2), feb, pp. 191–217. 12, 32

[50] Spearot, D. E., Jacob, K., and McDowell, D., 2007. "Dislocation nucleation from bicrystal interfaces with dissociated structure." *International Journal of Plasticity,* **23**(1), jan, pp. 143–160. 12, 32, 33

[51] Hertzberg, R., Vinci, R., and Hertzberg, J., 2012. *Deformation and Fracture Mechanics of Engineering Materials, 5th Edition*. Wiley. 21

[52] Schmid, E., and Boas, W., 1935. *Kristallplastizität*. Springer - Verlag Berlin Heidelberg. 24

[53] Yu, M.-h., 2002. "Advances in strength theories for materials under complex stress state in the 20th Century." *Applied Mechanics Reviews,* **55**(3), jun, pp. 169–218. 27

[54] Warner, R., 2012. *Applied Statistics: From Bivariate Through Multivariate Techniques: From Bivariate Through Multivariate Techniques*. SAGE Publications. 28

[55] McPhie, M., Berbenni, S., and Cherkaoui, M., 2012. "Activation energy for nucleation of partial dislocation from grain boundaries." *Computational Materials Science,* **62**, sep, pp. 169–174. 32

[56] Homer, E. R., Foiles, S. M., Holm, E. A., and Olmsted, D. L., 2013. "Phenomenology of

shear-coupled grain boundary motion in symmetric tilt and general grain boundaries." *Acta Materialia,* **61**(4), feb, pp. 1048–1060. 32

[57] Spearot, D. E., Jacob, K. I., and McDowell, D. L., 2005. "Nucleation of dislocations from [001] bicrystal interfaces in aluminum." *Acta Materialia,* **53**(13), aug, pp. 3579–3589. 33

[58] Roters, F., Eisenlohr, P., Hantcherli, L., Tjahjanto, D., Bieler, T., and Raabe, D., 2010. "Overview of constitutive laws, kinematics, homogenization and multiscale methods in crystal plasticity finite-element modeling: Theory, experiments, applications." *Acta Materialia,* **58**(4), feb, pp. 1152–1211. 37

APPENDIX A. CODE

A.1 Triaxial Stress States: MATLAB

The following will print triaxial stress states (λ unit vectors) to the terminal. These lines can be saved to a file called quadCubeSubdivide_4.txt and are then used in the super computing BASH script launch.sh. It uses a suite of functions from Anton Semechko [45] which is available on line at https://www.mathworks.com/matlabcentral/fileexchange/37004-suite-of-functions-to-perform-uniform-sampling-of-a-sphere as of 23-May-2016.

```
close all
clear all
clc

addpath('./S2 Sampling Toolbox/')

% Generate a quad cube mesh
fv=QuadCubeMesh;
for i=2:4
    fv=SubdivideSphericalMesh(fv,1);
end

% print the verticies that will be the Lambda vectors
fv.vertices
```

A.2 Super Computing Code: BASH Scripting

The super computing code presented here is current as of 24-May-2016.

A.2.1 Thermalization

The following code will thermalize the bicrystal and write it out as a restart file (consumable by LAMMPS) and a dump file (consumable by OVITO).

thermalize.sh:

```
#!/bin/bash

JOB_MSG=`
sbatch <<SBATCH_EOF
#!/bin/bash

#SBATCH --ntasks=24
#SBATCH --time=10:00:00   # walltime
#SBATCH --mem-per-cpu=2048M   # memory per CPU core
#SBATCH --output=./tmp/thermalize.$1.slurm
#SBATCH --gres=gpu:4

time mpirun /path/to/lammps/executable/lmp_gpu_exe <<THERMALIZE_EOF -log ./tmp
    /thermalize.log -sf gpu -pk gpu 4 -nocite

units metal
boundary p p p
atom_style atomic
pair_style eam/alloy

read_data ./ni.81.data
pair_style eam/alloy
pair_coeff * * ./ni1.set Ni Ni Ni Ni Ni
neighbor 2 bin

replicate 1 $1 $1

velocity all create 0.2 54321 dist gaussian
fix ensemble all npt temp 0.1 0.1 0.3 x 0 0 5 y 0 0 5 z 0 0 5 nreset 1
run 100000

write_data ./thermalized/ni.81.thermo.0p1.size.$1.data
THERMALIZE_EOF

SBATCH_EOF
`

JOB_ID=${JOB_MSG##* }

sbatch <<SBATCH_EOF
#!/bin/bash

#SBATCH --ntasks=1
#SBATCH --time=10:00:00   # walltime
#SBATCH --mem-per-cpu=2048M   # memory per CPU core
#SBATCH --output=./tmp/orig.$1.slurm
#SBATCH --depend=afterok:$JOB_ID
```

```

time mpirun /path/to/lammps/executable/lmp_exe <<DUMP -log ./tmp/orig.$1.log

units metal
boundary p p p
atom_style atomic
pair_style eam/alloy

# set up stress calculations
read_data ./thermalized/ni.81.thermo.0p1.size.$1.data
pair_style eam/alloy
pair_coeff * * ./ni1.set Ni Ni Ni Ni Ni
neighbor 2 bin

compute sv all slip/atom 0.01 3.52
compute cna all cna/atom 3.0046
dump dump_dislocations all custom 500 ./thermalized/ni.81.thermo.0p1.size.$1.
    dump id type x y z c_sv[1] c_sv[2] c_sv[3] c_cna

run 0

DUMP_EOF

SBATCH_EOF
```

A.2.2 Simulation

launch.sh: The following will submit jobs to run each simulation on the super computer. Serial or GPU accelerated submission scripts are possible.

```
#!/bin/bash

# this string specifies what to name the root folder and sub folders for the
    simulations
prefix=quad4_s1

script=./script.in
data_in=./path/to/inputfile/ni.81.thermo.0p1.size.1.data

cp $script /path/to/compute/staging/scripts/$prefix.in
cp $data_in /path/to/compute/staging/datafiles/$prefix.data

mkdir /path/to/compute/output/$prefix

# 'serial' can be changed to 'gpu' to use GPU accelerated hardware
awk -v exe_type=serial -v prefix=$prefix -v stress=-1000000 '{
```

```
        vectx=$1*stress
        vecty=$2*stress
        vectz=$3*stress
        vectx_pos=-1*$1*stress
        vecty_pos=-1*$2*stress
        vectz_pos=-1*$3*stress
        name= prefix _ vectx_pos _ vecty_pos _ vectz_pos
        cmd_line_args=-output=/path/to/comute/staging/slurms/ name .slurm
            -job-name= name  ./submit_ exe_type .sh  name   vecty   vectz
             vectx   prefix
        system(sbatch  cmd\_line\_args)
} ' /path/to/triaxial/stress/states/quadCubeSubdivide\_4.txt
```

submit_serial.sh: This submission script will run a single simulation serially (on a single processor). It can be used by setting the 'exe_type' variable in 'launch.sh' to 'serial'.

```
#!/bin/bash

#SBATCH --ntasks=1
#SBATCH --time=48:00:00   # walltime
#SBATCH --nodes=1
#SBATCH --mem-per-cpu=2048M   # memory per CPU core

name=$1
vecty=$2
vectz=$3
vectx=$4
prefix=$5

mkdir /path/to/compute/output/$prefix/$name
mkdir /path/to/compute/output/$prefix/$name/dumps/
mkdir /path/to/compute/output/$prefix/$name/thermovals/
mkdir /path/to/compute/output/$prefix/$name/others/

DIR=$( cd $( dirname $BASH_SOURCE[0] ) && pwd )
FILE=`basename $0`
DIRFILE=DIR/FILE
cp -a $DIRFILE /path/to/compute/output/$prefix/$name/others/submission_script.
    sh
cp -a /path/to/compute/staging/scripts/$prefix.in /path/to/compute/output/
    $prefix/$name/others/script.in
cp -a /path/to/compute/staging/datafiles/$prefix.data /path/to/compute/output/
    $prefix/$name/others/datafile.data

time /path/to/lammps/executable/lmp_exe \
        -log /path/to/compute/output/$prefix/$name/others/lammps.log \
        -in /path/to/compute/output/$prefix/$name/others/script.in \
        -var NAME $name \
```

```
        -var FORCE_X $vectx \
        -var FORCE_Y $vecty \
        -var FORCE_Z $vectz \
        -var PREFIX $prefix
```

submit_gpu.sh: This submission script will run a single simulation on GPU accelerated hardware. It can be used by setting the 'exe_type' variable in 'launch.sh' to 'gpu'.

```
#!/bin/bash

#SBATCH --ntasks=24
#SBATCH --gres=gpu:4
#SBATCH --time=3:00:00   # walltime
#SBATCH --exclusive
#SBATCH --mem-per-cpu=2048M   # memory per CPU core

name=$1
vecty=$2
vectz=$3
vectx=$4
prefix=$5

module load cuda/6.5.14

mkdir /path/to/compute/output/$prefix/$name
mkdir /path/to/compute/output/$prefix/$name/dumps/
mkdir /path/to/compute/output/$prefix/$name/thermovals/
mkdir /path/to/compute/output/$prefix/$name/others/

DIR=$( cd $( dirname $BASH_SOURCE[0] ) && pwd )
FILE=`basename $0`
DIRFILE=DIR/FILE

echo Look Here
echo $DIR
echo $FILE

cp -a $DIRFILE /path/to/compute/output/$prefix/$name/others/submission_script.
    sh
cp -a /path/to/compute/staging/scripts/$prefix.in /path/to/compute/output/
    $prefix/$name/others/script.in
cp -a /path/to/compute/staging/datafiles/$prefix.data /path/to/compute/output/
    $prefix/$name/others/datafile.data

time mpirun /path/to/lammps/executable/lmp_gpu_exe \
        -log /path/to/compute/output/$prefix/$name/others/lammps.log \
        -in /path/to/compute/output/$prefix/$name/others/script.in \
```

```
        -sf gpu -pk gpu 4 \
        -var NAME $name \
        -var FORCE_X $vectx \
        -var FORCE_Y $vecty \
        -var FORCE_Z $vectz \
        -var PREFIX $prefix
```

This LAMMPS input file instructs LAMMPS how to run a triaxial simulation. It calculates local stress and prints snap shots of the simulation periodically so we can later determine the slip system the GB nucleates on. It requires a LAMMPS executable with the optional VORONOI package and the custom compute_slip_atom.h and compute_slip_atom.cpp files (see A.2.3).

```
# initialization
units metal
boundary p p p
atom_style atomic
pair_style eam/alloy

# atom definition
read_data /path/to/compute/output/${PREFIX}/${NAME}/others/datafile.data
pair_style eam/alloy
pair_coeff * * ./ni1.set Ni Ni Ni Ni Ni
neighbor 2 bin

###########################################################################

variable lat equal 3.52
variable cnaparam equal 0.8536*v_lat

###########################################################################
# set up dump

compute cna all cna/atom ${cnaparam}
compute sv all slip/atom 0.01 ${lat}

variable isdefect atom (c_cna==2||c_cna==3||c_cna==4||c_cna==5)
group defect_atoms dynamic all var isdefect every 100

dump dump_dislocations defect_atoms custom 100 /path/to/compute/output/${
    PREFIX}/${NAME}/dumps/dump.* id type x y z c_sv[1] c_sv[2] c_sv[3] c_cna

###########################################################################
# set up kill criteria

variable ndefect equal count(defect_atoms)
variable natoms equal count(all)
```

```
34
35 variable stepNumber equal step
36
37 ###########################################################################
38 # this computes the standoff stress for 5-10 Angstroms away from the GB
39
40 region standoff_atoms_a5_region block $(63-10) $(63-5) INF INF INF INF
41 group standoff_atoms_a5 region standoff_atoms_a5_region
42 compute peratom_a5 standoff_atoms_a5 stress/atom NULL
43 compute p_a5 standoff_atoms_a5 reduce sum c_peratom_a5[1] c_peratom_a5[2]
       c_peratom_a5[3] c_peratom_a5[4] c_peratom_a5[5] c_peratom_a5[6]
44 compute vol_a5 standoff_atoms_a5 voronoi/atom
45 compute v_a5 standoff_atoms_a5 reduce sum c_vol_a5[1]
46 variable pxx_a5 equal 0.0001*c_p_a5[1]/c_v_a5
47 variable pyy_a5 equal 0.0001*c_p_a5[2]/c_v_a5
48 variable pzz_a5 equal 0.0001*c_p_a5[3]/c_v_a5
49 variable pxy_a5 equal 0.0001*c_p_a5[4]/c_v_a5
50 variable pxz_a5 equal 0.0001*c_p_a5[5]/c_v_a5
51 variable pyz_a5 equal 0.0001*c_p_a5[6]/c_v_a5
52
53 fix print_a5 all print 100 $stepNumber $pxx_a5 $pyy_a5 $pzz_a5 $pxy_a5 $pxz_a5
       $pyz_a5 file /path/to/compute/output/${PREFIX}/${NAME}/thermovals/
       thermovals.a.5 screen no title
54
55 region standoff_atoms_b5_region block $(63+5) $(63+10) INF INF INF INF
56 group standoff_atoms_b5 region standoff_atoms_b5_region
57 compute peratom_b5 standoff_atoms_b5 stress/atom NULL
58 compute p_b5 standoff_atoms_b5 reduce sum c_peratom_b5[1] c_peratom_b5[2]
       c_peratom_b5[3] c_peratom_b5[4] c_peratom_b5[5] c_peratom_b5[6]
59 compute vol_b5 standoff_atoms_b5 voronoi/atom
60 compute v_b5 standoff_atoms_b5 reduce sum c_vol_b5[1]
61 variable pxx_b5 equal 0.0001*c_p_b5[1]/c_v_b5
62 variable pyy_b5 equal 0.0001*c_p_b5[2]/c_v_b5
63 variable pzz_b5 equal 0.0001*c_p_b5[3]/c_v_b5
64 variable pxy_b5 equal 0.0001*c_p_b5[4]/c_v_b5
65 variable pxz_b5 equal 0.0001*c_p_b5[5]/c_v_b5
66 variable pyz_b5 equal 0.0001*c_p_b5[6]/c_v_b5
67
68 fix print_b5 all print 100 $stepNumber $pxx_b5 $pyy_b5 $pzz_b5 $pxy_b5 $pxz_b5
       $pyz_b5 file /path/to/compute/output/${PREFIX}/${NAME}/thermovals/
       thermovals.b.5 screen no title
69
70 ###########################################################################
71
72 compute mypress all pressure thermo_temp
73 variable pxx equal -0.0001*c_mypress[1]
74 variable pyy equal -0.0001*c_mypress[2]
75 variable pzz equal -0.0001*c_mypress[3]
```

```
variable pxy equal -0.0001*c_mypress[4]
variable pxz equal -0.0001*c_mypress[5]
variable pyz equal -0.0001*c_mypress[6]

fix print_global all print 100 $stepNumber $pxx $pyy $pzz $pxy $pxz
    $pyz file /path/to/compute/output/${PREFIX}/${NAME}/thermovals/thermovals.
    global screen no title

########################################################################

fix ensemble all npt temp 0.1 0.1 0.3 x 0 ${FORCE_X} 5 y 0 ${FORCE_Y} 5 z 0 ${
    FORCE_Z} 5 nreset 1

# this will kill the simulation when 35% of atoms are in non FCC
    configurations. That will be well after the relevant GB dislocation
    nucleation event has occurred.
variable nkill equal 0.35*v_natoms
run 1000000 start 0 stop 1000000 every 1000 if 'ndefect >nkill' then 'quit'
```

A.2.3 Slip Vector Compute Code

The following code implements a compute in LAMMPS to calculate the slip vector as proposed in [22]. It was emailed to me by Doctor Homer on 13-April-2015. For help compiling LAMMPS with a custom compute, see lammps.sandia.gov/doc/Section_modify.html (current as of 24-May-2016).

compute_slip_atom.h:

```
/* -*- c++ -*- ----------------------------------------------------------
   LAMMPS - Large-scale Atomic/Molecular Massively Parallel Simulator
   http://lammps.sandia.gov, Sandia National Laboratories
   Steve Plimpton, sjplimp@sandia.gov

   Adapted from compute_displace_atom and compute_centro_atom
   Written by Eric Homer and Garritt Tucker.
   Based on slip vector by J. A. Zimmerman, C. L. Kelchner, P. A. Klein,
      J. C. Hamilton, and S. M. Foiles. PRL, 2001, vol 87, 165507.

   Copyright (2003) Sandia Corporation.  Under the terms of Contract
   DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
   certain rights in this software.  This software is distributed under
   the GNU General Public License.

   See the README file in the top-level LAMMPS directory.
------------------------------------------------------------------------- */
```

```
18
19  #ifdef COMPUTE_CLASS
20
21  ComputeStyle(slip/atom,ComputeSlipAtom)
22
23  #else
24
25  #ifndef LMP_COMPUTE_SLIP_ATOM_H
26  #define LMP_COMPUTE_SLIP_ATOM_H
27
28  #include compute.h
29
30  namespace LAMMPS_NS {
31
32  class ComputeSlipAtom : public Compute {
33   public:
34    ComputeSlipAtom(class LAMMPS *, int, char **);
35    ~ComputeSlipAtom();
36    void init();
37    void init_list(int, class NeighList *);
38    void compute_peratom();
39    double memory_usage();
40
41  private:
42    int nmax;
43    double cutsq,btol;
44    class NeighList *list;
45    double **slip;
46    char *id_fix;
47    class FixStore *fix;
48
49
50  };
51
52  }
53
54  #endif
55  #endif
56
57  /* ERROR/WARNING messages:
58
59  E: Illegal ... command
60
61  Self-explanatory.  Check the input script syntax and compare to the
62  documentation for the command.  You can use -echo screen as a
63  command-line option when running LAMMPS to see the offending line.
64
```

```
E: Could not find compute displace/atom fix ID

Self-explanatory.

*/
```

compute_slip_atom.cpp:

```
/* ----------------------------------------------------------------------
   LAMMPS - Large-scale Atomic/Molecular Massively Parallel Simulator
   http://lammps.sandia.gov, Sandia National Laboratories
   Steve Plimpton, sjplimp@sandia.gov

   Adapted from compute_displace_atom and compute_centro_atom
   Written by Eric Homer and Garritt Tucker.
   Based on slip vector by J. A. Zimmerman, C. L. Kelchner, P. A. Klein,
      J. C. Hamilton, and S. M. Foiles. PRL, 2001, vol 87, 165507.

   Copyright (2003) Sandia Corporation.  Under the terms of Contract
   DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
   certain rights in this software.  This software is distributed under
   the GNU General Public License.

   See the README file in the top-level LAMMPS directory.
------------------------------------------------------------------------- */

#include math.h
#include string.h
#include compute_slip_atom.h
#include atom.h
#include update.h
#include group.h
#include domain.h
#include modify.h
#include fix.h
#include fix_store.h
#include memory.h
#include error.h
//#include stdlib.h
#include neighbor.h
#include neigh_list.h
#include neigh_request.h
#include force.h
#include pair.h
#include comm.h

using namespace LAMMPS_NS;

```

```
41 /* ---------------------------------------------------------------------- */
42
43 ComputeSlipAtom::ComputeSlipAtom(LAMMPS *lmp, int narg, char **arg) :
44   Compute(lmp, narg, arg)
45 {
46   if (narg < 4) error->all(FLERR,Illegal compute slip/atom command);
47
48   btol = force->numeric(FLERR,arg[3]);
49
50   //Set cutoff
51   if (narg == 5) {
52     cutsq = force->numeric(FLERR,arg[4]);
53     cutsq = cutsq * cutsq;
54   }
55   else cutsq = -1;
56
57   peratom_flag = 1;
58   size_peratom_cols = 4;
59
60   // create a new fix STORE style
61   // id = compute-ID + COMPUTE_STORE, fix group = compute group
62
63   int n = strlen(id) + strlen(_COMPUTE_STORE) + 1;
64   id_fix = new char[n];
65   strcpy(id_fix,id);
66   strcat(id_fix,_COMPUTE_STORE);
67
68   char **newarg = new char*[5];
69   newarg[0] = id_fix;
70   newarg[1] = group->names[igroup];
71   newarg[2] = (char *) STORE;
72   newarg[3] = (char *) 1;
73   newarg[4] = (char *) 3;
74   modify->add_fix(5,newarg);
75   fix = (FixStore *) modify->fix[modify->nfix-1];
76   delete [] newarg;
77
78   // calculate xu,yu,zu for fix store array
79   // skip if reset from restart file
80
81   if (fix->restart_reset) fix->restart_reset = 0;
82   else {
83     double **xoriginal = fix->astore;
84
85     double **x = atom->x;
86     int *mask = atom->mask;
87     int nlocal = atom->nlocal;
```

```
88
89      for (int i = 0; i < nlocal; i++)
90        if (mask[i] & groupbit)
91          for (int j = 0; j < 3; j++)
92            xoriginal[i][j] = x[i][j];
93        else xoriginal[i][0] = xoriginal[i][1] = xoriginal[i][2] = 0.0;
94    }
95
96    // per-atom slip array
97
98    nmax = 0;
99    slip = NULL;
100 }
101
102 /* ---------------------------------------------------------------------- */
103
104 ComputeSlipAtom::~ComputeSlipAtom()
105 {
106   // check nfix in case all fixes have already been deleted
107
108   if (modify->nfix) modify->delete_fix(id_fix);
109
110   delete [] id_fix;
111   memory->destroy(slip);
112 }
113
114 /* ---------------------------------------------------------------------- */
115
116 void ComputeSlipAtom::init()
117 {
118   // set fix which stores original atom coords
119
120   int ifix = modify->find_fix(id_fix);
121   if (ifix < 0) error->all(FLERR,Could not find compute slip/atom fix ID);
122   fix = (FixStore *) modify->fix[ifix];
123
124   // check details of pair_style and copies of this compute style
125
126   if (force->pair == NULL)
127     error->all(FLERR,Compute slip/atom requires a pair style be defined);
128
129   int count = 0;
130   for (int i = 0; i < modify->ncompute; i++)
131     if (strcmp(modify->compute[i]->style,slip/atom) == 0) count++;
132   if (count > 1 && comm->me == 0)
133     error->warning(FLERR,More than one compute slip/atom);
134
```

```
135   //check the cutoff value and set default if necessary
136   if (cutsq > force->pair->cutforce * force->pair->cutforce)
137     error->all(FLERR,Compute slip/atom:  invalid cutoff value);
138   if (cutsq  < 0) {
139     cutsq = force->pair->cutforce * force->pair->cutforce;
140   }
141
142   // need an occasional full neighbor list
143
144   int irequest = neighbor->request((void *) this);
145   neighbor->requests[irequest]->pair = 0;
146   neighbor->requests[irequest]->compute = 1;
147   neighbor->requests[irequest]->half = 0;
148   neighbor->requests[irequest]->full = 1;
149   neighbor->requests[irequest]->occasional = 1;
150 }
151
152 /* ---------------------------------------------------------------------- */
153
154 void ComputeSlipAtom::init_list(int id, NeighList *ptr)
155 {
156   list = ptr;
157 }
158
159 /* ---------------------------------------------------------------------- */
160
161 void ComputeSlipAtom::compute_peratom()
162 {
163
164   int i,j,k,ii,jj,inum,jnum;
165   double xtmp,ytmp,ztmp,delx,dely,delz,rsq;
166   int *ilist,*jlist,*numneigh,**firstneigh;
167
168   invoked_peratom = update->ntimestep;
169
170   // grow local slip array if necessary
171
172   if (atom->nlocal > nmax) {
173     memory->destroy(slip);
174     nmax = atom->nmax;
175     memory->create(slip,nmax,4,slip/atom:slip);
176     array_atom = slip;
177   }
178
179   // invoke full neighbor list (will copy or build if necessary)
180
181   neighbor->build_one(list);
```

```
182
183   inum = list->inum;
184   ilist = list->ilist;
185   numneigh = list->numneigh;
186   firstneigh = list->firstneigh;
187
188   // dx,dy,dz = slip of atom from original position
189   // original unwrapped position is stored by fix
190   // for triclinic, need to unwrap current atom coord via h matrix
191
192   // compute slip vector for each atom in group
193   // use full neighbor list
194
195   double **xoriginal = fix->astore;
196
197   double **x = atom->x;
198   int *mask = atom->mask;
199   imageint *image = atom->image;
200   int nlocal = atom->nlocal;
201
202   double del0[3],delf[3];
203
204   int ns;
205
206   for (ii = 0; ii < inum; ii++) {
207     i = ilist[ii];
208     for (k = 0; k < 4; k++) slip[i][k]=0.0;
209     ns = 0;
210     if (mask[i] & groupbit) {
211       jlist = firstneigh[i];
212       jnum = numneigh[i];
213
214       // loop over list of all neighbors within force cutoff
215
216       for (jj = 0; jj < jnum; jj++) {
217         j = jlist[jj];
218         j &= NEIGHMASK;
219
220         for (k = 0; k < 3; k++)
221           del0[k] = xoriginal[i][k] - xoriginal[j][k];
222         domain->minimum_image(del0);
223
224         rsq = 0;
225         for (k = 0; k < 3; k++)
226           rsq += del0[k] * del0[k];
227
228         if (rsq < cutsq) {
```

```
229
230           for (k = 0; k < 3; k++)
231             delf[k] = x[i][k] - x[j][k];
232           domain->minimum_image(delf);
233
234           double slipmag = 0;
235           double slipcheck[3];
236
237           for (k = 0; k < 3; k++) {
238             slipcheck[k] = del0[k] - delf[k];
239             slipmag += slipcheck[k] * slipcheck[k];
240           }
241
242           if (slipmag > btol) {
243             ns++;
244             for (k = 0; k < 3; k++)
245               slip[i][k] += slipcheck[k];
246           }
247         }
248       }
249
250       if (ns > 0) {
251         //calculate final division and magnitude
252         for (k = 0; k < 3; k++) {
253           slip[i][k] /= -ns;
254           slip[i][3] += slip[i][k] * slip[i][k];
255         }
256         slip[i][3] = sqrt(slip[i][3]);
257       }
258     }
259   }
260 }
261
262 /* ----------------------------------------------------------------------
263    memory usage of local atom-based array
264 ------------------------------------------------------------------------- */
265
266 double ComputeSlipAtom::memory_usage()
267 {
268   double bytes = nmax*4 * sizeof(double);
269   return bytes;
270 }
```

A.3 Post Processing Code: MATLAB

Post processing was performed in MATLAB. It was developed under MATLAB 7.12.0 (R2011a) on a Linux machine. This code is as it existed on 23-May-2016.

A.3.1 Data Structures

The Dump class is used to parse and store a LAMMPS dump file. The SetUp and getStandard classes build a structure that specifies things such as color and data layout. Both files are used throughout the other code.

```
classdef Dump < handle
        %DUMP Summary of this class goes here
        %   Detailed explanation goes here

        properties (GetAccess = public, SetAccess = immutable)
                location
                timestep
                xmin
                xmax
                ymin
                ymax
                zmin
                zmax
                numatoms
                atomproperties

                % made public for HOPEFULLY rare cases that some one needs
                    access
                % to this due to performance reasons.
                data_GENERALLY_PRIVATE
        end

        methods (Access = public)
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                % constructors

                function [ this ] = Dump(location)
                        if nargin==0
                                warning('Warning:Ricky:no_params',['No
                                    parameters were passed to Dump constructor
                                    . '...
                                        'It must be passed a ''location''
                                           parameter!']);
                                return
```

```
31                      end
32
33                      this.location=location;
34
35                      d=readdump_all(location);
36
37                      temp=textscan(strrep(d.atomheader,'ITEM: ATOMS ',''),'
                            %s');
38                      this.atomproperties=temp{1};
39
40                      this.timestep=d.timestep;
41                      this.numatoms=d.Natoms;
42                      this.xmin=d.x_bound(1);
43                      this.xmax=d.x_bound(2);
44                      this.ymin=d.y_bound(1);
45                      this.ymax=d.y_bound(2);
46                      this.zmin=d.z_bound(1);
47                      this.zmax=d.z_bound(2);
48
49                      this.data_GENERALLY_PRIVATE = d.atom_data;
50                  end
51
52                  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
53                  % methods
54
55                  function [ output ] = select(this,opperandProperties,
                        booleanTest,requestedProperties)
56                      % select - returns properties for rows that pass a
                            boolean test
57                      %
58                      % opperandProperties - these are the names of
                            properties that
59                      %   will be passed to the boolean test. They must be
                            values
60                      %   that exist in the class's atomproperties property.
61                      %
62                      % booleanTest - this is a function handle that
                            performs a
63                      %   boolean test on each row of atom data. It will be
                            passed
64                      %   the each row's opperandProperties in an array. For
65                      %   instance, if the booleanTest was
66                      %        @(arg) arg(1)>5 && arg(2)<7
67                      %   and the opperandProperties were
68                      %        {'x' 'c_sym'}
69                      %   then the method call would
```

```
70                      %   return all requesetedProperties (see below) for
                            rows that
71                      %   had an x value greater than 5 and a c_sym value
                            less than 7
72                      %
73                      % requestedProperties - these are the names of
                            properties that
74                      %   will be returned. A row will be returned with a
                            column for
75                      %   each property
76                      %
77                      % output - an [N,M] array where N = the number of rows
                             that
78                      %   passed the booleanTest and M = the number of
                            requested
79                      %   properties
80
81                      % get the properties that are to be opperated on
82                      opperandInds=this.getColIndicies(opperandProperties);
83                      d=this.data_GENERALLY_PRIVATE(:,opperandInds);
84
85                      % get the row indicies that passed the function's test
86                      ind=pass~=0;
87
88                      % get the requested properties for each row
89                      requestedInds=this.getColIndicies(requestedProperties)
                            ;
90                      output=this.data_GENERALLY_PRIVATE(ind,requestedInds);
91              end
92
93      end
94
95      methods (Access=private)
96              function [ indicies ] = getColIndicies(this,colnames)
97                      indicies = [];
98                      for index=1:length(this.atomproperties)
99                              [found,loc]=ismember(this.atomproperties(index
                                    ),colnames);
100                             if found
101                                     indicies(loc)=index; %#ok<AGROW>
102                             end
103                     end
104                     if length(indicies) ~= length(colnames)
105                             error('Cannot find one or more requested
                                    properties in dump file')
106                     end
107             end
```

```
        end
end
```

This is a wrapper for the get_activation_SYSTEMS.m code

```
function [s graphic_info]=SINGLE_ParseSimulation(criteria,dump_loc,letter,
    do_show)

if ~exist('do_show')
        do_show=false;
end

snap_shot_dump=Dump(dump_loc);
s=[];
[s graphic_info]=ParseGrainLetter(s,criteria,letter,snap_shot_dump,do_show);

end

function [s graphic_info]=ParseGrainLetter(s,criteria,grainLetter,
    snap_shot_dump,do_show)

% determine the dislocation content of the simulation after it nucleated
% Only use atoms within user defined ranges

if grainLetter=='a'
        evaluation_ids=criteria.orig.select({'x'},@(args) args(1) < 60 && args
            (1) > 50,{'id'});
else
        evaluation_ids=criteria.orig.select({'x'},@(args) args(1) > 66 && args
            (1) < 76,{'id'});
end

[mymap,~,~,graphic_info]=get_activation_SYSTEMS(snap_shot_dump,evaluation_ids,
    criteria.orig,criteria.pid,grainLetter,do_show,criteria);

% format mymap for better consumption
keys=mymap.keys;
for ii=1:length(mymap.keys)
        key=keys{ii};
        val=mymap(key);
        entries(ii,:)=[str2num(key) val]; %#ok<ST2NM>
end
entries=sortrows(entries,-7);
entries=[standardizeSlipSystem(entries(:,1:6)) entries(:,7)];

tmp=NaN;
primary_slip_sys=entries(1,1:6);
percent_primary=100*entries(1,7)/sum(entries(:,7));
```

```
for kk=1:size(criteria.slip_systems,1)
        if all(criteria.slip_systems(kk,:)==primary_slip_sys)
                tmp=kk;
                break
        end
end
if isnan(tmp)
        warning('WARNING:RICKY:UNKNOWN_SLIP_SYSTEM',['UNKNOWN SLIP SYSTEM
            ENCOUNTERED! ' num2str(primary_slip_sys) ' PLEASE ADD IT TO THE
            SetUp.m FILE!'])
        tmp=-1;
end
primary_nucleation_system_ind=tmp;

s.nucleation_content=entries;
s.primary_nucleation_system_ind=primary_nucleation_system_ind;
s.percent_content_on_primary=percent_primary;

end
```

This code will analyze a submitted Dump file object and determine the slip systems that have been activated within it. The program groups atoms by the dislocation they are on and then fits a plane normal to them. This is the slip plane. It also analyzes the atom's slip vector's [22] to determine each dislocation's slip direction. The slip plane and slip direction together constitute a slip system. The code does not respect periodic boundaries; however, was sufficiently robust for the program's needs.

```
function [ mydata colonyinfo fig graphic_info ]=get_activation_SYSTEMS(dump,
    dislocationids,referencedump,pid,grain_letter,doplot,criteria)

fig=[];

warning ('off','all');

[lab2grain grain2lab]=get_lab2grain_matrix(pid,grain_letter);

slipdirectioninfo_full=setup_slipdirectioninfo(grain2lab,true,true);
slipdirectioninfo_part=setup_slipdirectioninfo(grain2lab,false,true);

slipdirectioninfo=[slipdirectioninfo_full slipdirectioninfo_part];

slipplaneinfo=setup_slipplaneinfo(grain2lab);

dislocationids=sort(dislocationids);
```

```
atoms=dump.select({'id'},@(args) ismember(args(1),dislocationids),{'x','y','z','c_sv[1]','c_sv[2]','c_sv[3]','id'});

if ~isnumeric(referencedump)
        s=[(dump.xmax-dump.xmin)/(referencedump.xmax-referencedump.xmin) (dump.ymax-dump.ymin)/(referencedump.ymax-referencedump.ymin) (dump.zmax-dump.zmin)/(referencedump.zmax-referencedump.zmin)];
        for ii=1:size(atoms,1)
                aa=atoms(ii,:);
                atoms(ii,:)=[aa(1)*s(1) aa(2)*s(2) aa(3)*s(3) aa(4)*s(1) aa(5)*s(2) aa(6)*s(3) aa(7)];
        end
end

slipdirectionindicies=get_slipdirections(atoms(:,4:6),lab2grain,slipdirectioninfo);
% % if doplot
% %     fig=figure;
% %     [ax1 ax2]=show_slipdirections(atoms,slipdirectionindicies,slipdirectioninfo);
% % end

[colonies colonyslipdirectionindicies colonyatoms]=get_colonies(atoms,slipdirectionindicies,3,4);
% % if doplot
% %     disp(['NUM COLONIES ' num2str(numel(colonies))])
% %     ax3=show_colonies(colonyatoms,colonies);
% % end

colonyslipplaneindicies=get_colony_slipplanes(atoms,colonies,lab2grain,slipplaneinfo);

% % if doplot
% %     axs=[ax1,ax2,ax3,ax4];
% % % %         arrayfun(@(handle) view(handle,[0 -1 0]),axs);
% %     Link=linkprop(axs,{'CameraUpVector','CameraPosition','CameraTarget'});
% %     setappdata(gcf,'StoreTheLink',Link);
% % else
% %     fig=NaN;
% % end

% format data
colony_sizes=cellfun(@(args) length(args),colonies)';
colony_slipplanes=cell2mat(arrayfun(@(args) slipplaneinfo(args).grain.integer,colonyslipplaneindicies,'UniformOutput',false));
colony_slipdirections=cell2mat(arrayfun(@(args) slipdirectioninfo(args).grain.integer,colonyslipdirectionindicies,'UniformOutput',false));
```

```
54
55  val=standardizeSlipSystem([colony_slipplanes colony_slipdirections]);
56  colony_slipplanes=val(:,1:3);
57  colony_slipdirections=val(:,4:6);
58
59  graphic_info.atoms=atoms;
60  graphic_info.colonies=colonies;
61  graphic_info.colonyslipdirectionindicies=colonyslipdirectionindicies;
62  graphic_info.colonyslipplaneindicies=colonyslipplaneindicies;
63  graphic_info.slipdirectioninfo=slipdirectioninfo;
64  graphic_info.slipplaneinfo=slipplaneinfo;
65  graphic_info.colony_slipplanes=colony_slipplanes;
66  graphic_info.colony_slipdirections=colony_slipdirections;
67
68  figure
69  if doplot
70          ax4=show_processeddata(atoms,colonies,colonyslipdirectionindicies,
                colonyslipplaneindicies,...
71              slipdirectioninfo,slipplaneinfo,criteria,...
72              colony_slipplanes,colony_slipdirections);
73  end
74
75  dat=[colony_slipplanes colony_slipdirections colony_sizes];
76  dat=sortrows(dat,7);
77
78  dat=[dat dot(dat(:,1:3),dat(:,4:6),2)];
79
80  % bin data into slip systems
81  mydata=containers.Map('KeyType','char','ValueType','double');
82  for ii=1:size(dat,1)
83          slipsystem=num2str(dat(ii,1:6));
84          if ~mydata.isKey(slipsystem)
85                  mydata(slipsystem)=0;
86          end
87          mydata(slipsystem)=mydata(slipsystem)+dat(ii,7);
88  end
89
90  colonyinfo.colony_sizes=colony_sizes;
91  colonyinfo.colony_slipplanes=colony_slipplanes;
92  colonyinfo.colony_slipdirections=colony_slipdirections;
93
94  % DO NOT RETURN THIS!!! THEY ARE INDEXES WITHOUT MEANING!!!
95  % % colonyinfo.colonies=colonies;
96
97  colony_ids=cell(1,length(colonies));
98  for ii=1:length(colonies)
99          colony_ids{ii}=atoms(colonies{ii},7)';
```

```
end

colonyinfo.colony_ids=colony_ids;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [colonies colonyslipdirectionindicies atoms]=get_colonies(atoms,slipdirectionindicies,minatomsperplane,shellsize)

ind=1;
activatedslipdirectionindicies=unique(slipdirectionindicies);

colonyslipdirectionindicies=[];
colonies=[];

for ii=1:length(activatedslipdirectionindicies)

        candidates=find(slipdirectionindicies==activatedslipdirectionindicies(ii))';
        % %     length(candidates)

        while ~isempty(candidates)
                target=candidates(1);
                [colony candidates]=findcolony(atoms,target,candidates,target,shellsize);

                if length(colony)>=minatomsperplane
                        colonies{ind}=colony;
                        colonyslipdirectionindicies(ind)=activatedslipdirectionindicies(ii);
                        ind=ind+1;
                end
        end

end

colonyslipdirectionindicies=colonyslipdirectionindicies(:);

end

function [curcolony candidates]=findcolony(atoms,target,candidates,curcolony,shellsize)

shellindicies=findallinshell(atoms,target,candidates,shellsize);

```

```
curcolony=[curcolony shellindicies];
candidates=setdiff(candidates,shellindicies);

% recurse, break condition will be met when there are no shell indicies
for ii=1:length(shellindicies)
        [curcolony candidates]=findcolony(atoms,shellindicies(ii),candidates,
            curcolony,shellsize);
end

candidates=setdiff(candidates,curcolony);

curcolony=unique(curcolony);

end

function shellindicies=findallinshell(atoms,target,candidates,shellsize)
target_pos=atoms(target,1:3);

shellindicies=[];
for ii=1:length(candidates)
        if pdist([target_pos ; atoms(candidates(ii),1:3)])<shellsize
                shellindicies=[shellindicies candidates(ii)];
        end
end

end

function colony_slip_indices=get_colony_slipplanes(atoms,colonies,lab2grain,
    slipplaneinfo)

colony_slip_indices=zeros(length(colonies),1);
for ii=1:length(colonies)
        colonypositions=atoms(colonies{ii},1:3);
        slipplane=fitNormal(colonypositions,false);
        slipplane=lab2grain*slipplane(:);
        slipplane=slipplane/norm(slipplane);

        angs=zeros(1,length(slipplaneinfo));
        for jj=1:length(slipplaneinfo)
                slipplane2=slipplaneinfo(jj).grain.normalized;
                ang=acosd(dot(slipplane,slipplane2));
                ang=abs(ang);
                ang=min(abs([ang ang-180]));
                angs(jj)=ang;
        end
        [~,colony_slip_indices(ii)]=min(angs);
        % %     min(angs)
```

```
187 end
188
189 end
190
191 function [ slip_direction_indicies ] = get_slipdirections(
        atom_slip_directions_lab,lab2grain,slipdirectioninfo )
192
193 slip_direction_indicies=zeros(length(atom_slip_directions_lab),1);
194 for ii=1:length(atom_slip_directions_lab)
195
196         slipdir=atom_slip_directions_lab(ii,:);
197         slipdir=lab2grain*slipdir(:);
198         slipdir=slipdir/norm(slipdir);
199
200         angs=zeros(1,length(slipdirectioninfo));
201         for jj=1:length(slipdirectioninfo)
202                 slipdir2=slipdirectioninfo(jj).grain.normalized;
203                 ang=acosd(dot(slipdir,slipdir2));
204                 ang=abs(ang);
205                 % %             ang=min(abs([ang ang-180]));
206                 angs(jj)=ang;
207         end
208         [~,slip_direction_indicies(ii)]=min(angs);
209 end
210
211 end
212
213 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
214 % visualization functions
215
216 function [ax1 ax2]=show_slipdirections(atoms,slipdirectionindicies,
        slipdirectioninfo)
217
218 % plot the raw data
219 ax1=subplot(2,2,1);
220 hold on
221 scatter3(atoms(:,1),atoms(:,2),atoms(:,3));
222 quiver3(atoms(:,1),atoms(:,2),atoms(:,3),atoms(:,4),atoms(:,5),atoms(:,6));
223 hold off
224 axis equal
225
226 % plot the data with standardized slip directions
227 ax2=subplot(2,2,2);
228 uu=unique(slipdirectionindicies);
229 cc=jet(length(uu));
230 hold on
231 for ii=1:length(uu)
```

```
        goodindicies=find(slipdirectionindicies==uu(ii));

        scatter3(atoms(goodindicies,1),atoms(goodindicies,2),atoms(
            goodindicies,3),40,cc(ii,:));
        slipdirectionlab=slipdirectioninfo(uu(ii)).lab;
        slipdirs=[ones(length(goodindicies),1)*slipdirectionlab(1) ones(length
            (goodindicies),1)*slipdirectionlab(2) ones(length(goodindicies),1)
            *slipdirectionlab(3)];
        quiver3(atoms(goodindicies,1),atoms(goodindicies,2),atoms(goodindicies
            ,3),slipdirs(:,1),slipdirs(:,2),slipdirs(:,3));
end
hold off
axis equal

end

function [ ax1 ]=show_colonies(atoms,colonies)

figure
cc=hsv(length(colonies));
for ii=1:length(colonies)
        subplot(4,4,ii)
        scatter3(atoms(colonies{ii},1),atoms(colonies{ii},2),atoms(colonies{ii
            },3),40,cc(ii,:));
        view([0 -1 0])
end
ax1=123;

end

function [ax1]=show_processeddata( atoms,colonies,colonyslipdirectionindicies,
    ...
        colonyslipplaneindicies,slipdirectioninfo,slipplaneinfo,criteria,
            planeStuff,dirStuff )

cc=[
        31,120,180
        51,160,44
        227,26,28
        255,127,0
        106,61,154
        177,89,40
        166,206,227
        178,223,138
        251,154,153
        253,191,111
        202,178,214
```

```
        255,255,153
        ]/256;

ax1=figure('Position',[100 100 600 600]);
hold on
for ii=1:length(colonies)

        slip_sys=[planeStuff(ii,:) dirStuff(ii,:)];

        [a,b]=ismember(slip_sys,criteria.slip_systems,'rows');
        if a==0
                continue
        end

        cur=atoms(colonies{ii},1:3);



        center=mean(cur);

        uvw1=slipdirectioninfo(colonyslipdirectionindicies(ii)).lab;
        quiver3(center(1),center(2),center(3),uvw1(1),uvw1(2),uvw1(3));

        uvw2=slipplaneinfo(colonyslipplaneindicies(ii)).lab;
        quiver3(center(1),center(2),center(3),uvw2(1),uvw2(2),uvw2(3));

        scatter3(cur(:,1),cur(:,2),cur(:,3),40,cc(b,:));
        view([0 -1 0])

end
hold off
axis equal

end

function [ slipdirectioninfo ] = setup_slipdirectioninfo(grain2lab,full,
    keepsign)
%SETUP_SLIPDIRECTIONINFO Summary of this function goes here
%   Detailed explanation goes here
if full
        rawslipdirections=[
                0 1 1
                0 1 -1
                1 0 1
                1 0 -1
                1 1 0
                1 -1 0
```

```
                ];
else
        rawslipdirections=[
                2 1 1
                2 -1 1
                2 1 -1
                1 2 1
                -1 2 1
                1 2 -1
                1 1 2
                -1 1 2
                1 -1 2
                -2 1 1
                -2 -1 1
                -2 1 -1
                1 -2 1
                -1 -2 1
                1 -2 -1
                1 1 -2
                -1 1 -2
                1 -1 -2
                ];

end

if keepsign
        rawslipdirections=[rawslipdirections ; -rawslipdirections];
end

for ii=1:length(rawslipdirections)
        curdir=rawslipdirections(ii,:);
        slipdirectioninfo(ii).grain.integer=curdir;
        slipdirectioninfo(ii).grain.normalized=curdir/norm(curdir);
        slipdirectioninfo(ii).lab=(grain2lab*slipdirectioninfo(ii).grain.
            normalized(:))';
end

end

function slipplaneinfo=setup_slipplaneinfo(grain2lab)

rawslipplanes=[
        1 1 1
        -1 1 1
        1 -1 1
        1 1 -1
        ];
```

```
364
365 for ii=1:length(rawslipplanes)
366         curdir=rawslipplanes(ii,:);
367         slipplaneinfo(ii).grain.integer=curdir;
368         slipplaneinfo(ii).grain.normalized=curdir/norm(curdir);
369         slipplaneinfo(ii).lab=(grain2lab*slipplaneinfo(ii).grain.normalized(:)
                )';
370 end
371 end
```

This code changes slip systems into a standardized format. Specifically, it changes the slip direction in such a way that the slip system's that were nucleated on in this project would have a positive resolved shear stress. Slip systems that needed modification were determined by hand.

```
1  function slip_systems=standardizeSlipSystem(slip_systems)
2
3  slipplanes=slip_systems(:,1:3);
4  slipdirections=slip_systems(:,4:6);
5
6  signThing=sign(slipplanes(:,1));
7  for ii=1:length(signThing)
8          if signThing(ii)==0
9                  signThing(ii)=sign(slipplanes(ii,2));
10         end
11 end
12 slipplanes=[signThing signThing signThing].*slipplanes;
13
14 signThing=sign(slipdirections(:,1));
15 for ii=1:length(signThing)
16         if signThing(ii)==0
17                 signThing(ii)=sign(slipdirections(ii,2));
18         end
19 end
20 slipdirections=[signThing signThing signThing].*slipdirections;
21
22 slip_systems=[slipplanes slipdirections];
23
24 for ii=1:size(slip_systems,1)
25         if all(slip_systems(ii,4:6)==[2 1 -1])
26                 slip_systems(ii,4:6)=[-2 -1 1];
27         end
28
29         if all(slip_systems(ii,4:6)==[2 -1 1])
30                 slip_systems(ii,4:6)=[-2 1 -1];
31         end
32
33         if all(slip_systems(ii,4:6)==[1 1 -2])
```

```
                slip_systems(ii,4:6)=[-1 -1 2];
        end

        if all(slip_systems(ii,4:6)==[2 -1 -1])
                slip_systems(ii,4:6)=[-2 1 1];
        end
end

end
```

```
function criteria=SetUp()

prefix='quad_s1';

criteria.pid=81;

criteria.folder_pattern=        ['/path/to/where/files/to/analyze/are/winning_slipMeasure_a/*'];
criteria.file_name_format=      ['/path/to/supercomputer/output/' prefix '/%s/dumps/dump.%i'];
criteria.thermoval_pattern=     ['/path/to/supercomputer/output/' prefix '/%s/thermovals/'];

criteria.orig_loc='/path/to/bicrystal/ni.81.thermo.0p1.size.1.dump';

criteria.dumpsteps=0:100:1000000;

criteria.rel_slip_sys_inds_a=[ 8  9 10 11 12];
criteria.rel_slip_sys_inds_b=[12 11  7  9  8        1];

% for full version
% % criteria.rel_slip_sys_inds_a=[criteria.rel_slip_sys_inds_a setdiff(1:12, criteria.rel_slip_sys_inds_a)];
% % criteria.rel_slip_sys_inds_b=[criteria.rel_slip_sys_inds_b setdiff(1:12, criteria.rel_slip_sys_inds_b)];

criteria=getStandard(criteria);

end
```

```
function criteria=getStandard(criteria)

PROPS=containers.Map;
PROPS('id')=1;
PROPS('type')=2;
PROPS('x')=3;
PROPS('y')=4;
PROPS('z')=5;
```

```
PROPS('c_sv[1]')=6;
PROPS('c_sv[2]')=7;
PROPS('c_sv[3]')=8;

criteria.PROPS=PROPS;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

orig=Dump(criteria.orig_loc);
criteria.orig=orig;
orig_dat=orig.data_GENERALLY_PRIVATE;
criteria.orig_dat=orig_dat;

gb_midpoint=63;

measure_system_x_bounds=[7 11];

criteria.measureSlipSystemIds_b=sort(orig_dat(...
        orig_dat(:,PROPS('x'))<gb_midpoint-measure_system_x_bounds(1)&...
        orig_dat(:,PROPS('x'))>gb_midpoint-measure_system_x_bounds(2),...
        PROPS('id')));
criteria.isTimeToMeasureSlipSystem_b=sort(orig_dat(...
        orig_dat(:,PROPS('x'))<gb_midpoint-measure_system_x_bounds(2)&...
        orig_dat(:,PROPS('x'))>gb_midpoint-measure_system_x_bounds(2)-5,...
        PROPS('id')));

criteria.measureSlipSystemIds_a=sort(orig_dat(...
        orig_dat(:,PROPS('x'))>gb_midpoint+measure_system_x_bounds(1)&...
        orig_dat(:,PROPS('x'))<gb_midpoint+measure_system_x_bounds(2),...
        PROPS('id')));
criteria.isTimeToMeasureSlipSystem_a=sort(orig_dat(...
        orig_dat(:,PROPS('x'))>gb_midpoint+measure_system_x_bounds(2)&...
        orig_dat(:,PROPS('x'))<gb_midpoint+measure_system_x_bounds(2)+5,...
        PROPS('id')));

criteria.slip_systems=[
        1     1     1     2    -1    -1          % 1
        1     1     1     1    -2     1          % 2
        1     1     1     1     1    -2          % 3
        1    -1    -1     2     1     1          % 4
        1    -1    -1     1     2    -1          % 5
        1    -1    -1     1    -1                2       % 6
        1    -1     1    -2      -1              1       % 7
        1    -1     1     1     2     1          % 8
        1    -1     1     1    -1    -2          % 9
        1     1    -1    -2     1      -1        % 10
        1     1    -1     1    -2    -1          % 11
```

```
        1     1    -1     1     1     2         % 12
        ];
criteria.slip_systems=standardizeSlipSystem(criteria.slip_systems);

% Color scheme from: http://colorbrewer2.org/
% see http://colorbrewer2.org/?type=qualitative&scheme=Paired&n=12
% gathered 16-May-2016

criteria.slip_colors_a=[
        31,120,180
        51,160,44
        227,26,28
        255,127,0
        106,61,154
        177,89,40
        166,206,227
        178,223,138
        251,154,153
        253,191,111
        202,178,214
        255,255,153
        ]/256;
criteria.slip_colors_b=criteria.slip_colors_a;

end
```

This code specifies how the lab frame and each crystal frame relate to each other. It is specific to the bicrystal used in this project.

```
function [ lab2grain grain2lab ] = get_lab2grain_matrix( pid, crystal )

if pid~=81
        error('This code only works for PID=81')
end

grainB2lab=[0.8165    0.4082    0.4082
        0.5345   -0.2673   -0.8018
        -0.2182    0.8729   -0.4364];
lab2grainB=grainB2lab';

grainA2lab=[0.8165    0.4082    0.4082
        0.5345   -0.8018   -0.2673
        0.2182    0.4364   -0.8729];
lab2grainA=grainA2lab';

if strcmpi(crystal,'a')
        lab2grain=lab2grainA;
```

```
else
        lab2grain=lab2grainB;
end
grain2lab=lab2grain';

end
```

A.3.2 Check Utility

The following code was used to choose the correct simulation snap shot to determine the primary slip system.

```
% the grain to analyze
grain_letter='b';

% paths
folder_base='/path/to/supercomputer/output/';
winning_base=['winning_slipMeasure_' grain_letter '/'];
loser_base=['loser_slipMeasure_' grain_letter '/'];
twinned_base=['twinned_slipMeasure_' grain_letter '/'];
multi_base=['multi_slipMeasure_' grain_letter '/'];

% get folders
folders=dir(folder_base);
orig_loc='/path/to/bicrystal/ni.81.thermo.0p1.size.1.dump';
orig=Dump(orig_loc);

criteria=SetUp();

for ii=1:length(folders)

        folder=folders(ii);

        if strcmpi(folder.name,'.')||strcmpi(folder.name,'..')
                continue
        end

        checkUtility(folder.name,criteria,grain_letter);

end
```

```
function s=checkUtility(folder_name,criteria,letter)

[~,~,m]=regexp(folder_name,'_(-?\d+)_(-?\d+)_(-?\d+)');
m=m{1};
```

```
s.setx=str2num(folder_name(m(1,1):m(1,2))); %#ok<*ST2NM>
s.sety=str2num(folder_name(m(2,1):m(2,2)));
s.setz=str2num(folder_name(m(3,1):m(3,2)));
s.folder_name=folder_name;
s.short_name=[num2str(s.setx/100) '|' num2str(s.sety/100) '|' num2str(s.setz
    /100)];

entry=['thermovals.' letter '.5'];

name=entry;
bads=entry=='.';
name(bads)='_';

cur=load([sprintf(criteria.thermoval_pattern,folder_name) entry]);
vons=1/sqrt(2)*((cur(:,2)-cur(:,3)).^2+(cur(:,2)-cur(:,4)).^2+(cur(:,3)-cur
    (:,4)).^2+3*(cur(:,5).^2+cur(:,6).^2+cur(:,7).^2));

rel=vons;
adaptive_cutoff=0.1*max(rel(1:end/2));
[maxima,prominence]=getprominence(rel);
inds=find(prominence>adaptive_cutoff);
max_von_ind=maxima(inds(1));

% % cases=[-50 -3 0 3 5 10 50 0.25 0.5 0.75];
cases=[0 3 5 0.125 0.25 0.5 0.75 0.85];

sims=cell(size(cases));
parfor ii=1:length(cases)
        try
                % get dislocation content

                if cases(ii)<1&&cases(ii)~=0
                        nuc_ind=ceil(max_von_ind+(length(cur)-max_von_ind)*
                            cases(ii));
                else

                        nuc_ind=max_von_ind+cases(ii);
                end

                try
                        dump_step=cur(nuc_ind,1);
                catch
                        dump_step=cur(end);
                end

                dump_loc=sprintf(criteria.file_name_format,folder_name,
                    dump_step);
```

```
                [tmp graphic_info{ii}]=SINGLE_ParseSimulation(criteria,
                    dump_loc,letter,true);

                sims=s;
                sims.(['nucleation_content_' letter])=tmp.nucleation_content;
                sims.(['percent_content_on_primary_' letter])=tmp.
                    percent_content_on_primary;
                sims.(['primary_nucleation_system_ind_' letter])=tmp.
                    primary_nucleation_system_ind;
                sims.(['thermovals_' letter])=cur;
                sims.(['success_' letter])=true;
                graphic_info{ii}.dump_step=dump_step;

                dump_locs{ii}=dump_loc;

        catch err %#ok<NASGU>
                err
                sims=NaN;
                graphic_info{ii}=NaN;
                dump_locs{ii}=NaN;
        end

        if isstruct(sims)
                [~,slip_dat{ii}]=GetAnswer(sims,criteria,letter)
        else
                slip_dat{ii}=NaN;
        end

        mySims{ii}=sims;

end

f=figure('Name',folder_name,'Position',75+[0 0 1400 1400/2]);

ax1=subplot(2,8,1);
ax2=subplot(2,8,2);
ax3=subplot(2,8,3);
ax4=subplot(2,8,4);
ax5=subplot(2,8,5);
ax6=subplot(2,8,6);
ax7=subplot(2,8,7);
ax8=subplot(2,8,8);
ax9=subplot(2,8,9);
ax10=subplot(2,8,10);
ax11=subplot(2,8,11);
ax12=subplot(2,8,12);
ax13=subplot(2,8,13);
```

```
ax14=subplot(2,8,14);
ax15=subplot(2,8,15);
ax16=subplot(2,8,16);

checkUtility_plot(ax1,graphic_info{1},criteria);
checkUtility_plot(ax2,graphic_info{2},criteria);
checkUtility_plot(ax3,graphic_info{3},criteria);
checkUtility_plot(ax4,graphic_info{4},criteria);
checkUtility_plot(ax5,graphic_info{5},criteria);
checkUtility_plot(ax6,graphic_info{6},criteria);
checkUtility_plot(ax7,graphic_info{7},criteria);
checkUtility_plot(ax8,graphic_info{8},criteria);
checkUtility_graph(ax9,slip_dat{1},criteria);
checkUtility_graph(ax10,slip_dat{2},criteria);
checkUtility_graph(ax11,slip_dat{3},criteria);
checkUtility_graph(ax12,slip_dat{4},criteria);
checkUtility_graph(ax13,slip_dat{5},criteria);
checkUtility_graph(ax14,slip_dat{6},criteria);
checkUtility_graph(ax15,slip_dat{7},criteria);
checkUtility_graph(ax16,slip_dat{8},criteria);

num=inputdlg('GIVE NUMBER');
close(f);
try
        num=str2num(num{1});

        % copy correct snapshots to local storage
        system(['cp ' dump_locs{num(1)} ' ./winning_slipMeasure_' letter '/'
            folder_name])
        if length(num)>1
                system(['cp ' dump_locs{num(2)} ' ./secondary_slipMeasure_'
                    letter '/' folder_name])
        end

catch %#ok<CTCH>

        % track failed attemtps
        system(['touch ./loser_slipMeasure_' letter '/' folder_name])
end

end
```

```
function s=ParseSimulation(non_global_id,folder_name,criteria,letter)

[~,~,m]=regexp(folder_name,'_(-?\d+)_(-?\d+)_(-?\d+)');
m=m{1};
s.setx=str2num(folder_name(m(1,1):m(1,2))); %#ok<*ST2NM>
s.sety=str2num(folder_name(m(2,1):m(2,2)));
```

```
s.setz=str2num(folder_name(m(3,1):m(3,2)));
s.folder_name=folder_name;
s.short_name=[num2str(s.setx/100) '|' num2str(s.sety/100) '|' num2str(s.setz
    /100)];

entry=['thermovals.' letter '.5'];

name=entry;
bads=entry=='.';
name(bads)='_';

cur=load([sprintf(criteria.thermoval_pattern,folder_name) entry]);
vons=1/sqrt(2)*((cur(:,2)-cur(:,3)).^2+(cur(:,2)-cur(:,4)).^2+(cur(:,3)-cur
    (:,4)).^2+3*(cur(:,5).^2+cur(:,6).^2+cur(:,7).^2));

rel=vons;
adaptive_cutoff=0.1*max(rel(1:end/2));
[maxima,prominence]=getprominence(rel);
inds=find(prominence>adaptive_cutoff);
max_von_ind=maxima(inds(1));

figure
hold on
plot_index=maxima(inds(1));
plot(criteria.dumpsteps(1:length(vons)),vons)
plot(criteria.dumpsteps(max_von_ind),vons(plot_index),'*')
title(s.folder_name)
hold off

s.non_global_id=non_global_id;
s.(['nuc_content_' letter])=NaN;
s.(['primary_nucleation_system_ind_' letter])=NaN;
s.(['percent_content_on_primary_' letter])=NaN;
s.(['numatomsused_' letter])=0;
s.(['success_' letter])=false;

dump_loc=['winning_slipMeasure_' letter '/' folder_name];

tmp=SINGLE_ParseSimulation(criteria,dump_loc,letter);
primary_slip_system=tmp.primary_nucleation_system_ind;
percent_content_on_primary=tmp.percent_content_on_primary;
nuc_content=tmp.nucleation_content;
num_atoms_used=sum(nuc_content(:,end));

s.(['nuc_content_' letter])=nuc_content;
s.(['primary_nucleation_system_ind_' letter])=primary_slip_system;
s.(['percent_content_on_primary_' letter])=percent_content_on_primary;
```

```
s.(['numatomsused_' letter])=num_atoms_used;
s.(['success_' letter])=true;

end
```

```
function checkUtility_graph(ax,data,criteria)

if ~isstruct(data)||length(data.rel)==1
        return
end

dumpsteps=criteria.dumpsteps;

axes(ax)
hold on
plot(dumpsteps(1:length(data.rel)),data.rel)
plot(dumpsteps(data.nuc_ind),data.rel(data.nuc_ind),'*')
hold off

end
```

```
function checkUtility_plot( ax,graphicStuff,criteria )

if ~isstruct(graphicStuff)
        return
end

atoms=graphicStuff.atoms;
colonies=graphicStuff.colonies;
colonyslipdirectionindicies=graphicStuff.colonyslipdirectionindicies;
colonyslipplaneindicies=graphicStuff.colonyslipplaneindicies;
slipdirectioninfo=graphicStuff.slipdirectioninfo;
slipplaneinfo=graphicStuff.slipplaneinfo;
planeStuff=graphicStuff.colony_slipplanes;
dirStuff=graphicStuff.colony_slipdirections;

cc=[
        31,120,180
        51,160,44
        227,26,28
        255,127,0
        106,61,154
        177,89,40
        166,206,227
        178,223,138
        251,154,153
        253,191,111
        202,178,214
```

```
        255,255,153
        ]/256;

axes(ax) %#ok<MAXES>
hold on
for ii=1:length(colonies)

        slip_sys=[planeStuff(ii,:) dirStuff(ii,:)];

        [a,b]=ismember(slip_sys,criteria.slip_systems,'rows');
        if a==0
                color=[0.1 0.1 0.1];
        else
                color=cc(b,:);
        end

        cur=atoms(colonies{ii},1:3);

        center=mean(cur);

        uvw1=slipdirectioninfo(colonyslipdirectionindicies(ii)).lab;
        quiver3(center(1),center(2),center(3),uvw1(1),uvw1(2),uvw1(3));

        uvw2=slipplaneinfo(colonyslipplaneindicies(ii)).lab;
        quiver3(center(1),center(2),center(3),uvw2(1),uvw2(2),uvw2(3));

        scatter3(cur(:,1),cur(:,2),cur(:,3),40,color);
        view([0 -1 0])

end
hold off
axis equal
title(num2str(graphicStuff.dump_step))

end
```

A.3.3 Post Processing

This code will compute the nucleation slip system and resolved shear stresses needed for nucleation. It will produce many of the figures found through this thesis.

```

% main script

close all
drawnow
```

```
clearvars -except simulations_1 b_sims a_sims b_fits
clc

font_name='arial';

set(0,'DefaultAxesFontSize',18);
set(0,'defaultUicontrolFontName',font_name)
set(0,'defaultUitableFontName',font_name)
set(0,'defaultAxesFontName',font_name)
set(0,'defaultTextFontName',font_name)
set(0,'defaultUipanelFontName',font_name)

% the grain to analyze. This can be changed to 'b'
letter='a';

criteria=SetUp();

system(['rm slipMeasure_dump_' letter '/*']);
system(['rm nucleation_dump_' letter '/*']);

winning_folder_pattern=['/home/rdw53/Documents/week_20160516/
    winning_slipMeasure_' letter '/*'];
loser_folder_pattern=['/home/rdw53/Documents/week_20160516/loser_slipMeasure_'
     letter '/*'];
anomolous_folder_pattern=['/home/rdw53/Documents/week_20160516/
    anomolous_slipMeasure_' letter '/*'];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% find slip systems
if ~exist('simulations_1','var')
        winning_folders=dir(winning_folder_pattern);
        winning_folders=winning_folders(3:end);

        parfor ii=1:length(winning_folders)
                folder_name=winning_folders(ii).name;
                fprintf(['1 Working on: ' folder_name '\n'])

                % %             simulations_1{ii}=ParseSimulation(ii,
                    folder_name,criteria);
                simulations_1{ii}=ParseSimulation_vetted(ii,folder_name,
                    criteria,letter);

                fprintf(['\tFinished: ' folder_name '\n'])
        end

        loser_folders=dir(loser_folder_pattern);
```

```
48          loser_folders=loser_folders(3:end);
49          anomolous_folders=dir(anomolous_folder_pattern);
50          anomolous_folders=anomolous_folders(3:end);
51          bad_folders=[loser_folders anomolous_folders];
52          for ii=1:length(bad_folders)
53
54                  folder_name=bad_folders(ii).name;
55
56                  [~,~,m]=regexp(folder_name,'_(-?\d+)_(-?\d+)_(-?\d+)');
57                  m=m{1};
58
59                  s_bad.folder_name=folder_name;
60
61                  s_bad.setx=str2num(folder_name(m(1,1):m(1,2))); %#ok<*ST2NM>
62                  s_bad.sety=str2num(folder_name(m(2,1):m(2,2)));
63                  s_bad.setz=str2num(folder_name(m(3,1):m(3,2)));
64                  s_bad.(['success_' letter])=false;
65
66                  simulations_1{end+1}=s_bad;
67          end
68
69  end
70
71  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
72
73  % load thermoval data
74  if ~exist('simulations_2','var')
75          simulations_2=simulations_1;
76          thermoval_pattern=criteria.thermoval_pattern;
77          parfor ii=1:length(simulations_2)
78                  s=simulations_2{ii};
79                  fprintf(['2 Working on: ' s.folder_name '\n'])
80                  s.thermovals_a=load([sprintf(thermoval_pattern,s.folder_name)
                        'thermovals.a.5']);
81                  s.thermovals_b=load([sprintf(thermoval_pattern,s.folder_name)
                        'thermovals.b.5']);
82                  fprintf(['\tFinished: ' s.folder_name '\n'])
83                  simulations_2{ii}=s;
84          end
85  end
86
87  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
88
89  % load thermoval data
90  if ~exist('simulations_3','var')
91          simulations_3=simulations_2;
92          % %     for ii=26%length(simulations_3)
```

```
        parfor ii=1:length(simulations_3)
                s=simulations_3{ii};
                fprintf(['3 Working on: ' s.folder_name '\n'])
                s=GetAnswer(s,criteria,letter);
                fprintf(['\tFinished: ' s.folder_name '\n'])
                simulations_3{ii}=s;
        end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

simulations=simulations_3;
f_nucSystem=plot_system_map(simulations,letter,'color_',['Nucleation System in
     grain ' upper(letter)],false,false,['nucSystem_' upper(letter)]);
return
f_trssAtNuc=plot_system_map(simulations,letter,'trssAtNuc_',['*\tau_{rss}^''
    in grain ' upper(letter)],false,false,['trss_' letter],'GPa');

good_simulations=simulations(cellfun(@(s) s.(['success_' letter])&&ismember(s
    .(['primary_nucleation_system_ind_' letter]),criteria.(['
    rel_slip_sys_inds_' letter])),simulations));
[fits_normAndCoslip fig_handles statistics]=PlotCriticalTauVsNormAndCoslip(
    good_simulations,criteria,letter,['slipFits_' letter]);

print_coeffs(fits_normAndCoslip,statistics,letter,criteria,['./figures/params_
    ' letter '.txt'])

f_yield=plot_theoreticalYieldSurface(criteria,fits_normAndCoslip,letter,['
    yield_' letter]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% these lines compare actual nucleation against theoretical nucleation
% based on all slip systems. To use them, SetUp.m should be modified to
% consider all 12 slip systems instead of just the ones known to nucleate.
% getStandard.m should also be modified to have more colors for each slip
% system
simulations=simulations_3;
f_actual=plot_system_map(simulations,letter,'color_',['Actual Slip System in
    Grain ' upper(letter)],false,false,['schmidIsBad1_' letter]);
f_schmidTheory=plot_system_map(simulations,letter,'
    theoreticalSystemBasedOffOfMaxSchmidFactor_color_setStress_',sprintf(['
    Slip System according to Max Schmid Factor in Grain ' upper(letter)]),
    false,false,['schmidIsBad2_' letter]);
```

This code will compute resolved shear, normal, and co-slip stresses on a submitted stress tensor for requested slip systems

```
function [ resolvedTrss resolvedNorm resolvedCoslip ] = get_trss( SIG_crystal,
    slipsystems )
%GET_TRSS Summary of this function goes here
%   Detailed explanation goes here

% Archival code to derive sp13:
%{
syms dx dy dz yx yy yz nx ny nz
syms sxx sxy sxz syy syz szz
l=[dx dy dz;yx yy yz;nx ny nz]
s=[sxx sxy sxz;sxy syy syz;sxz syz szz]
lt=[dx yx nx;dy yy ny;dz yz nz]
sp=l*s*lt
sp(1,3)
sp(3,3)
%}

resolvedTrss=zeros(1,size(slipsystems,1));
resolvedNorm=zeros(1,size(slipsystems,1));
for slipsystemindex=1:size(slipsystems,1)

        slipsystem=slipsystems(slipsystemindex,:);

        n=[slipsystem(1) slipsystem(2) slipsystem(3)];
        n=n/norm(n);
        nx=n(1);
        ny=n(2);
        nz=n(3);

        d=[slipsystem(4) slipsystem(5) slipsystem(6)];
        d=d/norm(d);
        dx=d(1);
        dy=d(2);
        dz=d(3);

        y=cross(n,d);
        yx=y(1);
        yy=y(2);
        yz=y(3);

        sxx=SIG_crystal(1,1);
        sxy=SIG_crystal(2,1);   syy=SIG_crystal(2,2);
        sxz=SIG_crystal(3,1);   syz=SIG_crystal(3,2);   szz=SIG_crystal(3,3);

        %       sp13=abs(nx*(dx*sxx + dy*sxy + dz*sxz) + ny*(dx*sxy + dy*syy +
            dz*syz) + nz*(dx*sxz + dy*syz + dz*szz));
```

```
        sp13=nx*(dx*sxx + dy*sxy + dz*sxz) + ny*(dx*sxy + dy*syy + dz*syz) +
            nz*(dx*sxz + dy*syz + dz*szz);
        sp33=nx*(nx*sxx + ny*sxy + nz*sxz) + ny*(nx*sxy + ny*syy + nz*syz) +
            nz*(nx*sxz + ny*syz + nz*szz);
        sp23=nx*(sxx*yx + sxy*yy + sxz*yz) + ny*(sxy*yx + syy*yy + syz*yz) +
            nz*(sxz*yx + syz*yy + szz*yz);

        resolvedTrss(slipsystemindex)=sp13;
        resolvedNorm(slipsystemindex)=sp33;
        resolvedCoslip(slipsystemindex)=sp23;

end

end
```

```
function [X,Y]=cart2stereo(x,y,z,plane)
% original from Doctor Eric Homer, sent to Ricky Wyman in an email on 20160516

% cartesian to stereographic
%assuming x,y,z input is already normalized
if nargin < 4
        plane=3;
end
if plane==1 %projection of the y–z plane at x=0
        X=y.*(1./(x+1));
        Y=z.*(1./(x+1));
elseif plane==2 %projection of the x–z plane at y=0
        Y=x.*(1./(y+1));
        X=z.*(1./(y+1));
elseif plane==3 %projection of the x–y plane at z=0
        X=x.*(1./(z+1));
        Y=y.*(1./(z+1));
else
        error('plane must be equal to 1,2,3')
end
end
```

```
function [X,Y]=cart2stereo(x,y,z,plane)
% original from Doctor Eric Homer, sent to Ricky Wyman in an email on 20160516

% cartesian to stereographic
%assuming x,y,z input is already normalized
if nargin < 4
        plane=3;
end
if plane==1 %projection of the y–z plane at x=0
        X=y.*(1./(x+1));
        Y=z.*(1./(x+1));
```

```
elseif plane==2 %projection of the x-z plane at y=0
        Y=x.*(1./(y+1));
        X=z.*(1./(y+1));
elseif plane==3 %projection of the x-y plane at z=0
        X=x.*(1./(z+1));
        Y=y.*(1./(z+1));
else
        error('plane must be equal to 1,2,3')
end
end
```

This code analyzes each simulation after slip content and the primary nucleation slip system has been determined. It determines the resolved shear, normal, and co-slip stresses at the point of nucleation.

```
function [simulation dataToPlot]=GetAnswer(simulation,criteria,letter)
        [simulation dataToPlot]=getAnswer(simulation,criteria,letter);
end

function [simulation dataToPlot]=getAnswer(simulation,criteria,grain_letter)

if ~simulation.(['success_' grain_letter]);
        nuc_ind=1;
        dataToPlot.rel=NaN;
        dataToPlot.nuc_ind=nuc_ind;
        return
end

lab2grain=get_lab2grain_matrix(criteria.pid,grain_letter);
primary_slipsys_ind=simulation.(['primary_nucleation_system_ind_' grain_letter
    ]);
rel_slipsys_inds=criteria.(['rel_slip_sys_inds_' grain_letter]);

thermovals=simulation.(['thermovals_' grain_letter]);
for ii=1:size(thermovals,1)

        entry=thermovals(ii,2:7);

        STRESS_lab=[
                entry(1) entry(4) entry(5)
                entry(4) entry(2) entry(6)
                entry(5) entry(6) entry(3)
                ];
        STRESS_grain{ii}=lab2grain*STRESS_lab*lab2grain';
        [sp13s(ii,:) sp33s(ii,:) sp23s(ii,:)]=get_trss(STRESS_grain{ii},
            criteria.slip_systems(rel_slipsys_inds,:));

```

```
        vonMises_stress(ii)=1/sqrt(2)*sqrt(...
                (entry(1)-entry(2))^2+...
                (entry(1)-entry(3))^2+...
                (entry(2)-entry(3))^2+...
                6*(entry(4)^2+entry(5)^2+entry(6)^2));

end

nucleation_system_index=rel_slipsys_inds==primary_slipsys_ind;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% change these if wanted

if all(nucleation_system_index==0)
        simulation.(['success_' grain_letter])=false;
        nuc_ind=1;
        dataToPlot.rel=NaN;
        dataToPlot.nuc_ind=nuc_ind;
else
        rel=sp13s(:,nucleation_system_index);
        adaptive_cutoff=0.2*max(rel(1:end/2));
        [maxima,prominence]=getprominence(rel);
        inds=find(prominence>adaptive_cutoff);
        nuc_ind=maxima(inds(1));

        dump_step=criteria.dumpsteps(nuc_ind);
        dump_loc=sprintf(criteria.file_name_format,simulation.folder_name,
            dump_step);
        system(['cp ' dump_loc ' nucleation_dump_' grain_letter '/' simulation
            .folder_name '__' num2str(dump_step)]);

        dataToPlot.rel=rel;
        dataToPlot.nuc_ind=nuc_ind;

end

simulation.(['nucInd_' grain_letter])=nuc_ind;
simulation.(['trss_dat_' grain_letter])=sp13s(:,nucleation_system_index);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
try
        setStressAtNuc_lab=350000*nuc_ind/numel(criteria.dumpsteps)*[
            simulation.setx 0 0;0 simulation.sety 0;0 0 simulation.setz];
catch
        setStressAtNuc_lab=350000*1/numel(criteria.dumpsteps)*[simulation.setx
             0 0;0 simulation.sety 0;0 0 simulation.setz];
end
```

```
setStressAtNuc_grain=lab2grain*setStressAtNuc_lab*lab2grain';
actual=get_trss(setStressAtNuc_grain,criteria.slip_systems(rel_slipsys_inds,:)
    );
[~,tmp]=max(abs(actual));
simulation.(['theoreticalSystemBasedOffOf'...
'MaxSchmidFactor_color_setStress_' grain_letter])=criteria.(['slip_colors_'
    grain_letter])(tmp,:);

try
        stressAtNuc_grain=STRESS_grain{nuc_ind};
        actual=get_trss(stressAtNuc_grain,criteria.slip_systems(
            rel_slipsys_inds,:));

        [~,tmp]=max(abs(actual));
        simulation.(['theoreticalSystemBasedOffOf'...
    'MaxSchmidFactor_color_actualStress_' grain_letter])=criteria.(['
       slip_colors_' grain_letter])(tmp,:);
catch
        simulation.(['theoreticalSystemBasedOffOf'...'
    'MaxSchmidFactor_color_actualStress_' grain_letter])=NaN;
end

simulation.(['trss_' grain_letter])=sp13s;
simulation.(['norm_' grain_letter])=sp33s;
simulation.(['cosl_' grain_letter])=sp23s;

simulation.(['trssAtNuc_' grain_letter])=sp13s(nuc_ind,nucleation_system_index
    );

color=criteria.(['slip_colors_' grain_letter])(primary_slipsys_ind==
    rel_slipsys_inds,:);
if isempty(color)
        color=[NaN NaN NaN];
end
simulation.(['color_' grain_letter])=color;

end
```

This code plots a requested simulation property on a stereographic image as a function of λ vector components. It was used to generate Figures 3.2 and 4.7

```
function f=plot_system_map(simulations,grain_letter,property,title_str,
    doannotate,specify_color,name,include_colorbar)

if ~exist('doannotate','var')
        doannotate=true;
end
```

```
6
7  if ~exist('specify_color','var')
8          specify_color=true;
9  end
10
11 if ~exist('include_colorbar','var')
12         axis_offset=[0 0 0 0];
13 else
14         axis_offset=[0 0.2 0 0];
15 end
16
17 bar2gpa=0.0001;
18
19 bads=[];
20 for ii=1:length(simulations)
21         s=simulations{ii};
22
23         label{ii}=s.folder_name;
24
25         xset(ii)=s.setx*bar2gpa;
26         yset(ii)=s.sety*bar2gpa;
27         zset(ii)=s.setz*bar2gpa;
28
29         try
30                 tmp=s.([property grain_letter]);
31                 if any(isnan(tmp))
32                         THROW_ERROR;
33                 end
34                 c(ii,:)=tmp;
35         catch %#ok<CTCH>
36                 if specify_color
37                         c(ii,:)=[0.4 0.4 0.4];
38                 else
39                         c(ii,:)=0;
40                 end
41                 bads=[bads ii];
42         end
43 end
44 xval=xset(:)/max(abs(xset));
45 yval=yset(:)/max(abs(yset));
46 zval=zset(:)/max(abs(zset));
47
48 figure_size=[3.25 2];
49 middle_x_offset=[figure_size(1)/2 0 0 0];
50
51 f=figure('Units','Inches','Resize','off','Position',[2 2 0 0]+[0 0 figure_size
       ],'Color',[1 1 1]);
```

```
52
53  font_size=12
54  x_margin_offset=[(font_size+2)/72.272 0 0 0];
55  % % y_margin
56
57  outer_multiplier=1.2;
58  for isfront=[true false]
59
60          good=xval>=0;
61          if ~isfront
62                  good=~good;
63          end
64
65          curx=xval(good);
66          cury=yval(good);
67          curz=zval(good);
68          curc=c(good,:);
69          [cury,curz]=cart2stereo(cury,curz,abs(curx));
70
71          dt=DelaunayTri([cury(:) curz(:)]);
72          circumcenters=dt.circumcenters;
73          ti=dt.vertexAttachments;
74
75          if isfront
76                  axis_side=(figure_size(1)-9*x_margin_offset(1))/2;
77                  axes('Units','Inches','Position',axis_offset+4*x_margin_offset
                        +[0 0 axis_side axis_side])
78                  scaling=1/(outer_multiplier*max(max(circumcenters)));
79          else
80                  axes('Units','Inches','Position',axis_offset+9*x_margin_offset
                        +[axis_side 0 0 0]+[0 0 axis_side axis_side])
81                  scaling=1/(outer_multiplier*max(max(circumcenters)));
82          end
83
84          adaptive_max=max(sqrt(cury.^2+curz.^2));
85
86          hold on
87          for ii=1:length(cury)
88
89                  cur_ti=ti{ii};
90
91                  x=circumcenters(cur_ti,1);
92                  y=circumcenters(cur_ti,2);
93                  if norm([cury(ii) curz(ii)])>0.95*adaptive_max
94                          [th,R]=cart2pol(x,y);
95                          [supx,supy]=pol2cart(th,outer_multiplier*R);
96
```

```
                        x=[x ; supx];
                        y=[y ; supy];

                        K=convhull(x,y);
                        x=x(K);
                        y=y(K);

                end
                patch(scaling*x,scaling*y,curc(ii,:));

        end
        hold off

        if isfront
                title('Tensile \sigma_x','FontSize',font_size)
        else
                title('Compressive \sigma_x','FontSize',font_size)
        end
        xlabel('\lambda_y','FontSize',font_size)
        ylabel('\lambda_z','FontSize',font_size)
        xlim([-1 1])
        ylim([-1 1])
        axis square

end

saveas(f,['./figures/' name])

end
```

This code makes scatter plots of the resolved shear, normal, and co-slip stresses on each slip system at the point of nucleation. It fits a plane to the simulations that nucleated on each plot's slip system. It returns these fitting parameters for future use. It was used to generate the sub figures in Figure 3.4

```
function [fits_normAndCoslip fig_handles statistics]=
    PlotCriticalTauVsNormAndCoslip(simulations,criteria,grain_letter,base_name
    )

target_slip_indicies=criteria.(['rel_slip_sys_inds_' grain_letter]);

for ii=1:length(target_slip_indicies)
        cur_slip_index=target_slip_indicies(ii);
        fits_normAndCoslip{ii,1}=cur_slip_index;
        [fits_normAndCoslip{ii,2} fig_handles(ii) statistics(ii)]=doPlotThing(
            criteria,simulations,grain_letter,ii,cur_slip_index,base_name);
```

```
9  end
10
11 end
12
13 function [fit_normAndCoslip fig_handle statistics]=doPlotThing(criteria,
       simulations,grainLetter,ind,cur_slip_index,base_name)
14
15 cur_slip_system=criteria.slip_systems(cur_slip_index,:);
16 cur_slip_system_printer=num2str(cur_slip_system,'%i');
17 cur_slip_system_latex=['(' num2str(cur_slip_system(1:3)) ')_{<' num2str(
       cur_slip_system(4:6)) '>}'];
18
19 cur_slip_system_color=criteria.(['slip_colors_' grainLetter])(ind,:);
20
21 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
22 % find out what simulations nucleated on what slip systems and group
23 % accordingly. Gather nucleation point information
24 xval=[];
25 yval=[];
26 zval=[];
27 for ii=1:length(simulations) %#ok<FORPF>
28
29         s=simulations{ii};
30
31         if cur_slip_index==s.(['primary_nucleation_system_ind_' grainLetter])
32                 good(ii)=true;
33                 nucleation_marker_size=600;
34         else
35                 good(ii)=false;
36                 nucleation_marker_size=50;
37         end
38
39         trss=s.(['trss_' grainLetter])(:,ind);
40         norms=s.(['norm_' grainLetter])(:,ind);
41         cosls=s.(['cosl_' grainLetter])(:,ind);
42
43         nuc_ind=s.(['nucInd_' grainLetter]);
44
45         xval(ii)=norms(nuc_ind);
46         yval(ii)=cosls(nuc_ind);
47         zval(ii)=trss(nuc_ind);
48
49         nucleation_marker_sizes(ii)=nucleation_marker_size;
50
51         nuc_color(ii,:)=s.(['color_' grainLetter]);
52
53         label{ii}=s.folder_name;
```

```
end

fig_handle=figure('Name',cur_slip_system_latex,'Position',100+[0 0 500 500],'
    Color',[1 1 1]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plot and get fit for normal and coslip stress

hold on
scatter3(xval(~good),yval(~good),zval(~good),nucleation_marker_sizes(~good),
    nuc_color(~good,:),'*');
scatter3(xval(good),yval(good),zval(good),nucleation_marker_sizes(good),
    nuc_color(good,:),'.');

try
        [fit_normAndCoslip G]=fit([xval(good)' yval(good)'],zval(good)','
            poly11');
        h=plot(fit_normAndCoslip);
        set(h,'FaceAlpha',0.5,'FaceColor',0.5*cur_slip_system_color,'EdgeColor
            ','None');
        title(sprintf([cur_slip_system_latex '\nR^2=' num2str(G.rsquare)]),'
            Color',cur_slip_system_color)
        statistics.R=G.rsquare;
catch %#ok<CTCH>
        title('Could not do a fit')
        fit_normAndCoslip=[];
        statistics.R=NaN;
end

statistics.N=sum(good);

xlabel('\sigma\prime_{rns} (GPa)')
ylabel('\tau\prime_{rco} (GPa)')
zlabel('\tau\prime_{rss} (GPa)')
set(gca,'XColor',cur_slip_system_color);
set(gca,'YColor',cur_slip_system_color);
set(gca,'ZColor',cur_slip_system_color);
view(115,10)
axis tight
zoom(0.8)

hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

saveas(fig_handle,['./figures/' base_name '_' cur_slip_system_printer]);

```

```
end
```

This code prints the coefficients that are used in tables [?], [?], [?], and [?].

```
function print_coeffs(fits_normAndCoslip,statistics,grain_letter,criteria,
    file_loc)

fid=fopen(file_loc,'w')

fprintf(fid,['Fitting parameters for Grain ' upper(grain_letter) ': (version
    1)\n'])
for ii=1:size(fits_normAndCoslip,1)
        ss=criteria.slip_systems(fits_normAndCoslip{ii,1},:);
        try
                str=['   Slip System=(' num2str(ss(1:3)) ')<' num2str(ss(4:6))
                     '>\ttau_crss=' num2str(fits_normAndCoslip{ii,2}.p00) '\
                    tsigma_crns=' num2str(-fits_normAndCoslip{ii,2}.p00/
                    fits_normAndCoslip{ii,2}.p10) '\ttau_crcs=' num2str(-
                    fits_normAndCoslip{ii,2}.p00/fits_normAndCoslip{ii,2}.p01)
                     '\tR=' num2str(statistics(ii).R) '\tN=' num2str(
                    statistics(ii).N) '\n'];
        catch %#ok<CTCH>
                str=['   Slip System=(' num2str(ss(1:3)) ')<' num2str(ss(4:6))
                     '>\tNO FIT POSSIBLE\t\n'];
        end
        fprintf(fid,str)
end
fprintf(fid,['Fitting parameters for Grain ' upper(grain_letter) ': (version
    2)\n'])
for ii=1:size(fits_normAndCoslip,1)
        ss=criteria.slip_systems(fits_normAndCoslip{ii,1},:);
        try
                str=['   Slip System=(' num2str(ss(1:3)) ')<' num2str(ss(4:6))
                     '>\ttau_crss=' num2str(fits_normAndCoslip{ii,2}.p00)...
                        '\ttau_crss/sigma_crns=' num2str(fits_normAndCoslip{ii
                            ,2}.p10) '\ttau_crss/tau_crcs=' num2str(
                            fits_normAndCoslip{ii,2}.p01)...
                        '\tR=' num2str(statistics(ii).R) '\tN=' num2str(
                            statistics(ii).N) '\n'];
        catch %#ok<CTCH>
                str=['   Slip System=(' num2str(ss(1:3)) ')<' num2str(ss(4:6))
                     '>\tNO FIT POSSIBLE\t\n'];
        end
        fprintf(fid,str)
end

fclose(fid);

```

```
end
```

This code uses the fitting coefficients to build a yield surface in principal stress space (with the principal stresses oriented on $\hat{x}$, $\hat{y}$, and $\hat{z}$. It was used to generate Figure 4.1

```
function f=plot_theoreticalYieldSurface(criteria,nucleation_criteria,
    grain_letter,save_name)

ind=1;

targets=1:length(nucleation_criteria);

for ii=1:length(targets);

        targ=targets(ii);

        nucleation_slip_ind=nucleation_criteria{targ,1};
        nucleation_criterion=nucleation_criteria{targ,2};

        if isempty(nucleation_criterion)
                continue
        end

        slip_system=criteria.slip_systems(nucleation_slip_ind,:);

        % build a function handle that reports if a particular stress state
        % would nucleate on the current fit
        yieldCriterion=@(ps) did_nucleate(...
                criteria.pid,ps,nucleation_criterion,slip_system,grain_letter)
                    ;

        % find three points on the surface of each nucleation constraint. Do
        % this by searching in three different directions (if going forward on
        % one direction fails, then look backwards. One of the two should
        % always work)
        p1=getPrincipalsAndActivation(yieldCriterion,0,0);
        if any(isnan(p1))
                p1=getPrincipalsAndActivation(yieldCriterion,0,0+pi);
        end
        p2=getPrincipalsAndActivation(yieldCriterion,0,pi/2);
        if any(isnan(p2))
                p2=getPrincipalsAndActivation(yieldCriterion,0,pi/2+pi);
        end
        p3=getPrincipalsAndActivation(yieldCriterion,1,0);
        if any(isnan(p3))
                p3=getPrincipalsAndActivation(yieldCriterion,1e5,0+pi);
        end
```

```
41
42          % build a matrix of three points lying on the surface of the current
43          % nucleation criterion
44          point_table=[p1';p2';p3'];
45
46          % find the normal to the plane
47          abc=fitNormal(point_table,false);
48
49          % find the intercept of the plane
50          d=dot(abc,p1);
51
52          % solve for the plane equation. This represents an algebraic equation
53          % of the current nucleation criterion
54          planeeqs(ind,:)=[abc(:)' d];
55          plane_colors(ind,:)=criteria.(['slip_colors_' grain_letter])(criteria
                .(['rel_slip_sys_inds_' grain_letter])==nucleation_slip_ind,:);
56          ind=ind+1;
57  end
58
59  % see http://www.mathworks.com/matlabcentral/answers/82901-how-to-plot
60  % -feasible-space-in-3d-after-subtracting-multiple-inequalities
61  % answer by Steven Grob gathered on 2-May-2016
62
63  % density
64  n=0.25;
65  low=-50;
66  high=70;
67  vals=low:n:high;
68
69  % create coordinates
70  [xx,yy,zz] = meshgrid(vals,vals,vals);
71
72  region3=boolean(ones(size(xx)));
73  for ii=1:size(planeeqs,1)
74          rr=planeeqs(ii,1)*xx+planeeqs(ii,2)*yy+planeeqs(ii,3)*zz;
75
76          what_side=0<planeeqs(ii,4);
77
78          if what_side
79                  reg{ii}=rr < planeeqs(ii,4);
80                  region3 = rr < planeeqs(ii,4) & region3;
81          else
82                  reg{ii}=rr > planeeqs(ii,4);
83                  region3 = rr > planeeqs(ii,4) & region3;
84          end
85  end
86
```

```
f=figure;
hold on
fv=isosurface(vals,vals,vals,region3,0.5);
for ii=1:length(reg)
        my_fv=isosurface(vals,vals,vals,reg{ii},0.5);
        points=intersect(my_fv.vertices,fv.vertices,'rows');

        if isempty(points)
                continue
        end

        conv_inds=convhull(points(:,1),points(:,2));
        h=patch(points(conv_inds,1),points(conv_inds,2),points(conv_inds,3),
            plane_colors(ii,:));
        set(h,'FaceAlpha',0.5)
end
hold off
xlim([-50 50])
ylim([-50 50])
zlim([-50 50])

% plot a line of hydrostatic stress
extreme_max=max([xlim ylim zlim]);
extrema=[0 extreme_max];
h=line(extrema,extrema,extrema);
set(h,'Color','k')
view(50,50)
xlabel('\sigma_x (GPa)')
ylabel('\sigma_y (GPa)')
zlabel('\sigma_z (GPa)')
hold off
grid on

% % print(f,['./figures/' save_name],'-dpng');
saveas(f,['./figures/' save_name]);

end

function [principals r]=getPrincipalsAndActivation(yield_criterion,pidist,
    theta)
% finds what principal stresses would cause nucleation to occur searching
% outwards from the piplane at  particular pi coordinate (specified by
% pdist) and direction (specified by theta).
%
% This is a brute force model. There are many ways it could be improved
% including computational (use a binary search instead of searching
% outwards in increments) and analytical (algebraic intersection of the
```

```
% search direction with the yield criteria)

r=0;
ind=0;

rot2piSpace=[
        1       1       1
        1       -1      1
        1       0       -2
        ];

while ind<1000

        principals=rot2piSpace*[pidist;r*sin(theta);r*cos(theta)];

        did_nucleate=yield_criterion(principals);
        if did_nucleate
                return
        end
        r=r+1e-1;
        ind=ind+1;
end

% nucleation did not occur. Return NaN values to communicate this
principals=NaN(3,1);

end

function bool_did_nuc=did_nucleate(pid,ps,nucleation_criterion,slip_system,
    grain_letter)

SIG_crystal=[ps(1) 0 0;0 ps(2) 0;0 0 ps(3)];

lab2grain=get_lab2grain_matrix(pid,grain_letter);
SIG_crystal=lab2grain*SIG_crystal*lab2grain';

[trss normal_stress cosl]=get_trss(SIG_crystal,slip_system);

try
        theory_trss=feval(nucleation_criterion,normal_stress,cosl);
catch %#ok<CTCH>
        theory_trss=feval(nucleation_criterion,normal_stress);
end

deviation=theory_trss-trss;

bool_did_nuc=deviation<0;
```

```
178
179 end
```