UNIVERSITY OF CALIFORNIA

Los Angeles

The Influence of Crystal Surfaces on Dislocation Behavior in Mesoscopic Plasticity: A Combined Dislocation Dynamics - Finite Element Approach

A thesis submitted in partial satisfaction

of the requirements for the degree Master of Science

in Mechanical Engineering

by

Rudolph Duran Martinez Jr.

2000

The thesis of Rudolph Duran Martinez Jr. is approved.

Gregory P. Carman

Vijay Gupta

Nasr M. Ghoniem, Committee Chair

University of California, Los Angeles

2000

For Kris, who’s undying support made this project a success.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

Many individuals deserve recognition for the time, effort, wisdom, and patience that they bestowed upon me. First, I would like to thank the members of the “Dislocations Team”. Professor Ghoniem is the engine that drives the lab. He is an able mentor, and provided suggestions and vision at critical times during the course of the project. Dr. Shahram Sharafat served as a sounding board for ideas, and a troubleshooter capable of advising me on how to overcome many tricky obstacles. Jianming Huang’s efforts were responsible for the Deformation portion of the Microplasticity code. We sat for many hours writing code and running analyses. He was patient, even when we navigated in circles. Dr. Shihhsi Tong was always willing to help with questions concerning the Fortran code. Thank you all for lending me your expertise.

Additionally, I would like to thank Professor Gupta and Professor Carman for their efforts as my nominating committee.

Lastly, I would like to thank Professor Behzad Bavarian for granting me the opportunity of working in his lab as an undergraduate researcher. This experience set the wheels in motion that made this graduate experience possible for me.

ABSTRACT OF THE THESIS

The Influence of Crystal Surfaces on Dislocation Behavior In Mesoscopic Plasticity:

A Combined Dislocation Dynamics - Finite Element Approach

By

Rudolph Duran Martinez Jr.

Master of Science in Mechanical Engineering

University of California, Los Angeles

2000

Professor Nasr M. Ghoniem, Chair

The research presented in this thesis focuses on the direct coupling of Dislocation Dynamics (DD) computer simulations with Finite Element Method (FEM) models to simulate plastic deformation of micro-scale structures. This involves a series of three-dimensional (3-d) DD simulations of BCC single crystals with a single shear loop in the (101) [-111] slip system, contained within planar boundaries under the influence of tensile loading. The purpose of these simulations is to explore the relationship between loop force distributions and the proximity of the loop to the boundary. Traction free boundary conditions on a single crystal model are satisfied through the superposition of the “image” stress field, and the elastic stress field.

With the traction free boundary condition satisfied, the force distribution on a shear loop is verified. This distribution consists of the superposition of Peierls, image, applied, and self forces. Force distributions are explored as a function of loop proximity to the boundary of the single crystal model. The deformation of the loop under the influence of these force distributions is computed using the Galerkin method, and the equilibrium geometry is plotted. Additionally, the deformation of a Frank-Reed (FR) source in a single crystal model under the influence of image forces, an applied stress, and Peierls forces with varying screw/edge force ratios is computed and plotted.

The results of these numerical analyses indicates that image forces play a significant role in dislocation force distributions and deformation to a depth from the surface which is directly proportional to the loop radius. Large out-of-plane image force distributions on closed loops in “oblique” slip plane/free surface orientations are verified. These forces act in such a way as to repel loop motion from the intersection of the slip plane with the free surface, while causing deformation through the mechanism of cross-slip. Numerical results also verify that image forces repel dislocations from free surfaces for normal slip plane/free surface orientations. Expansion or contraction of shear loops is dependent on the critical applied stress, the radius of curvature, and the proximity/orientation of the loop with respect to the boundary.

1 INTRODUCTION AND OBJECTIVE

1.1 An Introduction to Mesoscale Modeling

Plastic deformation of materials is receiving ever-increasing attention by the research community. This attention is driven by three issues: 1.) Materials deformation beyond the elastic regime is described by constitutive equations of continuum plasticity for which stress is related to displacements or strains within the material. Because these constitutive equations are empirical, an extensive experimental database is required for design and analysis. This is a limitation toward technological progress in many materials dependent applications. 2.) Recent technological advancements that are allowing for the miniaturization of mechanical and electrical systems have brought about the discovery that the size of a component has a significant influence on strength. Smaller means stronger [28]. These effects have been readily observed in indentation testing of thin films and torque testing of copper wire [28]. 3.) The last issue driving exploration into the fundamental aspects of materials deformation is the longstanding observation that the deformation of materials is inherently inhomogeneous. Localized regions of deformation surrounded by material with little or no deformation are observed in metallic alloys [11]. Scale effects, the heterogeneity of material deformation and the desire to model the deformation of materials for which no constitutive equations exists, has prompted exploration and research into the foundations of plastic deformation [33].

This exploration consists of experimental observations and numerical simulations. Experimental observations are achieved through the use of atomic force microscopy (AFM), transmission electron microscopy (TEM), x-ray diffraction, internal friction measurements, and other microstructural analysis techniques. Mesoscale modeling is the tool by which researchers numerically simulate the macroscopic behavior associated with plastic deformation. This modeling approach is the link between first-principles based interatomic potentials at the atomic scale, and continuum mechanics at the macroscale. Mesoscale modeling is dominated by the theories and computational methods known as Dislocation Dynamics (DD). Dislocation Dynamics describes the stress-strain behavior, mobility and interaction of dislocations in crystal structures [34,37]. This methodology was first proposed over 12 years ago by Ghoniem and Amodeo [29], and Lepinoux and Kubin [30] for 2-d dislocation models.

Since then, 3-d simulations have gradually been developed, with the ultimate goal of modeling materials behavior under extreme conditions for which experimental validation is difficult to obtain [31]. The 1992 U.S. moratorium on nuclear testing has forced the U. S. Department of Energy (DOE) to explore and develop these simulation techniques in order to predict materials properties in the aging nuclear stockpile, and thus ensure the reliability of these nuclear weapons [32,42]. To handle the enormous number of computations required by these simulations, the DOE began sponsoring the Accelerated Strategic Computing Initiative (ASCI) in 1995. IBM has built the fastest computer in the world for this program. The ASCI "White" supercomputer will perform 12.3 trillion operations per second, thus making it the ultimate mesoscale modeling tool.

The research presented in this thesis seeks to advance our understanding of the effects of free crystal surfaces on the dynamics of shear dislocation loops through the use of a coupled mesoscale-finite element modeling approach. Although it is commonly accepted that surface boundaries affect dislocation behavior by inducing image stress fields, which decay in the material volume with distance from the free surface, the precise behavior of dislocations within this stress field is not well understood. Examples of the importance of dislocation dynamics and the interaction of dislocations with free surfaces are given below. Following this section the dislocation model used in this research is briefly introduced.

1.2 Significance And Importance Of Surface Effects On Plastic Deformation

Dislocation interaction with free surfaces is of great importance in the understanding of a number of phenomena that govern the mechanical response of materials to applied loads. For example, the interaction between dislocations and free surfaces plays a significant role in fatigue behavior. In single crystals, it is widely accepted that surface roughness caused by the formation of persistent slip bands (PSBs) is the dominant contributing factor to fatigue crack initiation [9-11,41]. Surface roughness is characterized by the protrusion and extrusion of slip planes due to the effects of dislocation motion and their interaction with the surface of a material. Fatigue crack nucleation often accounts for a large portion of the fatigue life of a component. Computer models, which simulate the surface roughening of various kinds of single crystals due to random slip on primary slip planes have accurately predicted the number of cycles required for crack nucleation [9,10]. It is believed that inclusions are the nucleation site for fatigue cracking. A micrograph of extrusions and inclusions, which characterize the PSB, is depicted below.

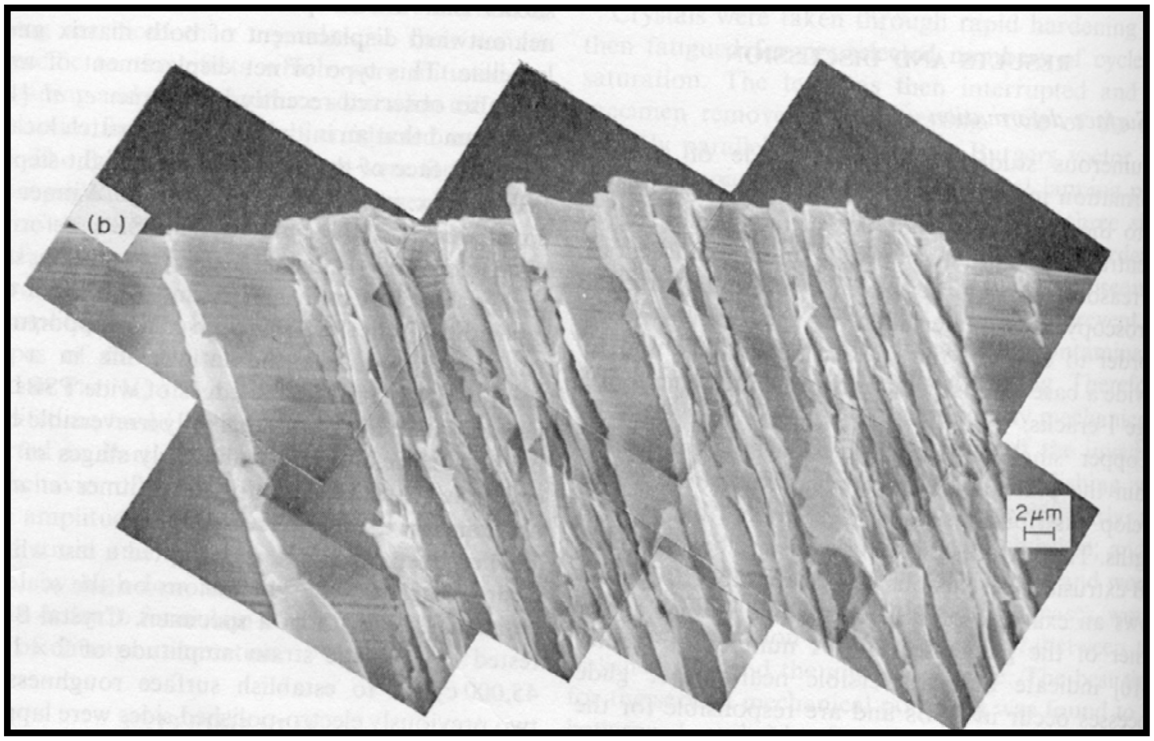


Figure 1.2.1: Surface roughness due to PSB/surface interaction in a copper single crystal, which was fatigue tested at a strain amplitude of 2 x10^{-3} for 120,000 cycles [12, p.328].

Dislocation-free surface interactions are of paramount importance to the reliability and performance of microelectronics as well. Wang and Lee have investigated the effect of a screw dislocation near a subsurface crack in silicon single crystal [21]. Machining processes used on silicon wafers often produce damage to the surface and subsurface layers. This damage involves cracking and the nucleation of dislocations, which leads to detrimental effects such as the diffusion of impurities, and electron-hole recombination. Image forces and stresses are critical to the understanding of these phenomena. Their simulation [21] indicates that for certain Burgers vector orientations, and screw dislocation positions, the stress intensity factor and strain energy associated with the crack are enhanced by the effects of the screw dislocation and the induced image forces.

Furthermore, the importance of dislocations and image stress fields can be seen in other micro-electronics applications utilizing thin films. The interaction between threading dislocations and free surfaces of a strained layer bonded to a substrate plays a significant role in the estimation of the critical thickness of a strained layer for a particular mismatch strain. Misfit dislocations in epitaxially grown thin films are known to be detrimental to the electrical properties of these materials [24].

The importance of dislocation-free surface interaction is visible in the field of micro-tribology with respect to MEMS component behavior and reliability [22,23,26]. A micromechanical dislocation model of frictional slip between two asperities presented by Hurtado and Kim [22] suggests that for contact areas smaller than a critical value, friction stress is constant and on the order of theoretical shear strength. In this model it is proposed that slip between two asperities is assisted by the nucleation and gliding of a dislocation loop. The dominant forces effecting the dislocations in this model are self forces, image forces, and applied forces. Free surfaces and image forces are being increasingly recognized as dominant factors in the mechanics of adhesion, wear, friction, and lubrication.

The last of many examples of the importance of dislocations and free surface effects involves an important experimental procedure: nanoindentation. Nanoindentation has become increasingly popular for the experimental characterization of material properties of thin films [27]. Because this technique is characterized by small loads and small geometric scales, deformation is dependent on the nucleation and interaction of dislocations. These events are strongly affected by factors such as surface and grain effects.

With all of the increased attention being paid to numerical material modeling at the mesoscale, DD is playing an increasingly dominant role in predicting materials behavior. Image stresses developed at the surface of crystal boundaries are now being recognized as an important element in unlocking the mystery of the motion and interaction of dislocations in the crystal lattice.

1.3 Thesis Objectives

The primary focus of this thesis is the direct coupling of Dislocation Dynamics computer simulations with FEM models to simulate plastic deformation of micro-scale structures. This involves a series of DD simulations of single crystals containing planar boundaries.

These simulations are made using Fortran computer codes developed in part by the author, and by researchers at UCLA under the direction of Professor Nasr Ghoniem. The primary computer program used in the modeling efforts described in this thesis is called "Microplasticity". The purpose of simulations made with this computer code is to quantify the relationship between loop force distributions, and the proximity of the loop to the boundary. Free surface boundary conditions on a single crystal model are satisfied through the superposition of the image stress field computed by the ANSYS finite element software, and the elastic stress field computed by Microplasticity.

With the free surface boundary condition satisfied, the force distribution on a shear loop in a single crystal with an applied tensile stress is verified. This force distribution consists of the superposition of Peierls, image, applied, and self forces. Force distributions are then explored as a function of loop proximity to the boundary of the single crystal model.

Having verified the force distribution on the dislocation loop, the deformation of the loop is computed and the equilibrium geometry is plotted. Additionally, the deformation of a Frank-Reed source in a single crystal model under the influence of an applied stress is computed and plotted.

Several assumptions were made in the simulations presented. Because the simulations take place at room temperature, climb deformation is ignored. Only the movement of dislocations on the slip plane is considered. For all loop analyses, only a single shear loop is considered. The crystal lattice is taken as being free of point defects and the materials behavior is taken to be isotropic. 10 and 20 mesh divisions per side, and 20 and 60 nodes per loop were compared for numerical accuracy considerations.

Chapter 2, “Status of Research on Surface Effects” details experimental and theoretical aspects that have been recently published. In Chapter 3, “The Method of Dislocation Dynamics”, we describe the mathematics behind the method of dislocation dynamics. Chapter 4, “The Proposed FEM/Dislocation Dynamics Approach” discusses the assumptions made and the computational approach used. Chapter 5, “Studies of Loop Force Distributions and Deformation Near Surfaces in BCC Metals” describes the results of the analyses conducted. Finally, conclusions and recommendations are given in Chapter 6.

Dislocations come in all shapes and sizes. Bulk and surface techniques have been developed which have allowed experimentalists to document dislocations in various crystal structures [50]. This introductory chapter would not be complete without some form of experimental evidence of dislocation loops. The following micrograph is offered as physical evidence of the simulated dislocations presented in this thesis.

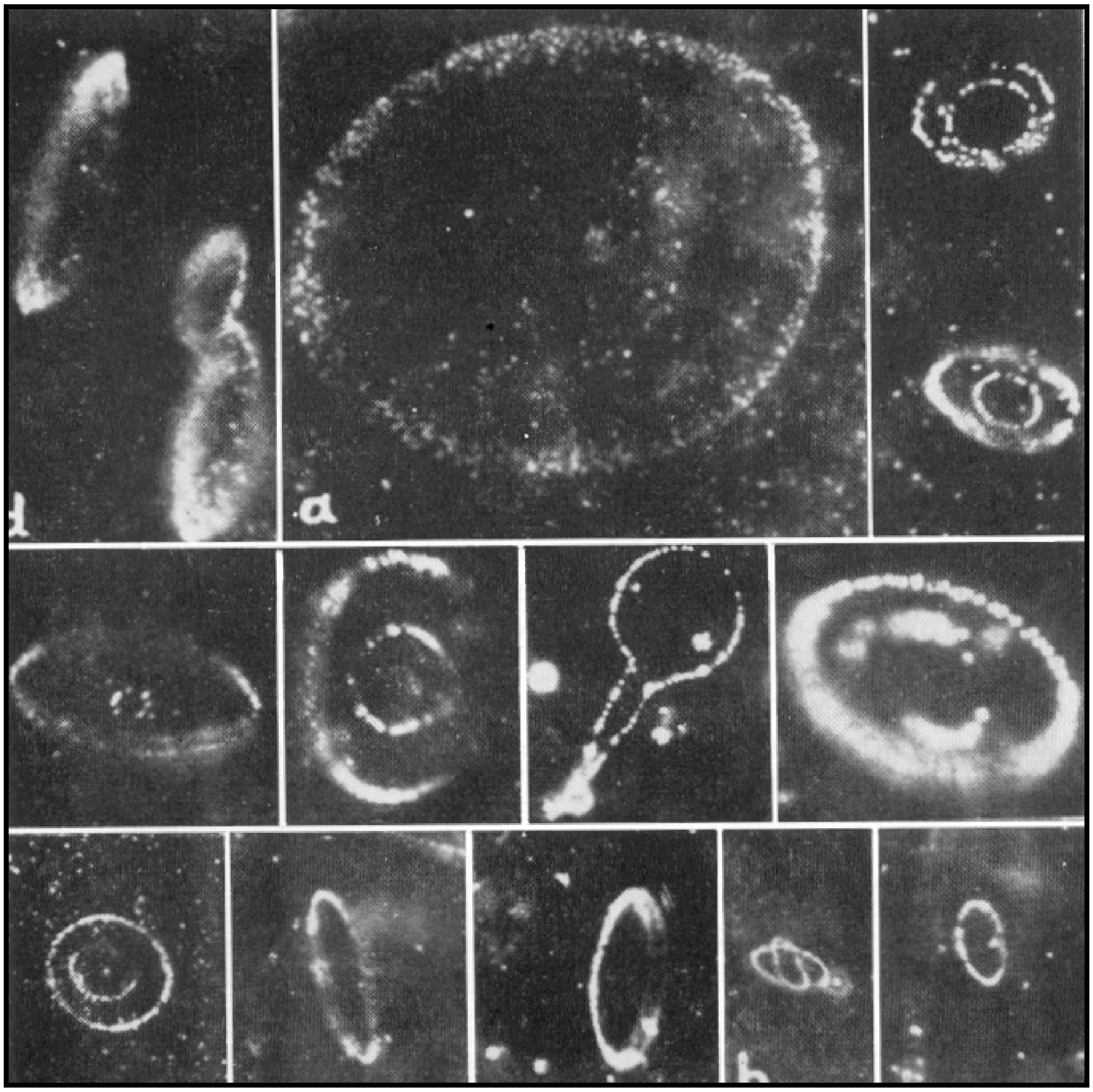


Figure 1.2.2: Micrographs of prismatic dislocation loops in potassium chloride [50].

2 STATUS OF RESEARCH ON SURFACE EFFECTS

A literature pursuit was conducted to gain a sense of the approach used by other researchers who are simulating similar Dislocation Dynamics based models. In this chapter, the key features of various studies on the experimental, theoretical and modeling related achievement of researchers in the field of Dislocations Dynamics and surface effects are described. Most of the modeling efforts uncovered by this literature review were geared toward either determining the hardening behavior of a single crystal, or qualifying and/or quantifying some microstructural phenomena relating to experimental observations.

Generally, as in the approach of Kubin, DeVincre and coworkers [36], dislocation lines are discretized into straight segments of either screw or edge character. The translation and rotation of each segment is numerically integrated according to the effective force computed at the midspan of the dislocation segment. The velocity of the dislocation is computed by a viscous mobility law. Rules relating to the dislocation core properties are established that attempt to account for dipole formation and cross slip.

Generally, single crystal boundary effects are treated in one of four different ways. The "periodic" boundary condition allows for dislocations to travel out of the unit cell on a given slip plane and return on the opposite side of the crystal in the same slip plane.

Another boundary condition that is more appropriate for describing the effects of single crystal surfaces is the free surface condition [36]. This boundary condition is employed in this thesis. Another method involves a "quasi-free" boundary condition [35]. In this methodology, the dislocation is only allowed to leave the unit cell under investigation. This is meant to simulate part of a bulk crystal. Other models neglect surface effects under the assumption that the image stress field decays rapidly. Thus, the image stress field is not significant within the depth of the model. Statistical data is recorded only within a sphere of material within the crystal [37].

No study uncovered by this literature review provides details on the image force distribution on dislocation loops in three dimensions. Most of the current methods which deal with free surface effects are approximate for 3-d simulations, or are based on 2-d models [44]. Now we will review the details of various DD based studies.

2.1 Single Crystal Modeling

Fivel, Gosling and Canova explore the image stress field associated with a 3-d single crystal model in "Implementing Image Stresses in a 3D Dislocation Simulation" (1996) [35]. This study focused on FCC copper. The scale of the model is deduced through the observations that screw segments cannot coexist within 500 Å of each other, and edge dislocations cannot coexist within 30 Å of each other. Thus the scale is defined as 26 Å.

Six Burgers vectors of (110) type are considered. Dislocations are described as having finite length segments of either edge or screw composition. These dislocation segments are restricted to glide or climb in certain specific directions. The overall dislocation moves or evolves based upon the individual movement of the pieces that compose it. These pieces behave according to established rules. These rules attempt to account for annihilation and evolution of dislocation geometry.

The stress field in the crystal is the superposition of the applied stress, the image stress and the elastic stress field from other dislocation loops. Stress is calculated only at midpoints on dislocation segments. Glide is determined through the following linear viscous law:

$$\tau * b = B\ v \tag{2.1.1}$$

where τ is the resolved shear stress, b is the Burgers vector, v is the velocity and B is the drag coefficient. Cross-slip is treated as a probabilistic event, and depends on the local stress and temperature. The Peierls stress represents lattice friction to dislocation motion, and it must be overcome for slip to occur.

The image stress field is calculated based on a finite element scheme with an adaptive mesh that increases with increasing surface traction field gradient. In this way, subtleties in the image stress field can be realized through high-resolution numerical analysis. For a single crystal model, image stresses are computed and displayed as a function of distance from the free surface for a single square dislocation loop, and for many loops.

Tang, Kubin and Canova present a BCC crystal plasticity simulation in "Dislocation Mobility and the Mechanical Response of BCC Single Crystals: A Mesoscopic Approach" (1997) [37]. This paper details the thermal nucleation of double kinks as the controlling mechanism in the mobility of screw dislocations in Tantalum. The temperature and strain rate dependence of the yield strength are explored. The methodology employed in this simulation involves straight dislocation segments of screw or edge type moving in the {110}<111> slip system. Cross slip occurs based on the probability of double kink nucleation. Stresses involved do not include a contribution from the image stresses induced at the surface. Only a sphere of material in the center of the crystal is considered. Velocity of edge components is related to the local stress, while the velocity of screw components is based on an Arrehenius law. Movement is limited by obstacles like forest dislocations. The velocity of the screw dislocations is governed by a stress dependent activation enthalpy equation, which involved numerically fitted experimental data.

Results for a single FR source are presented. Under a stress on the order of the yield stress in FCC metals, the dislocation expands by producing long screw components. These screw components are sessile at low stresses. When the stress reaches the macroscopic yield stress (50 MPa), the screw components become mobile and begin expanding. Kinks are produced, generating more edge segments. The entire process produces long screw components, which are stated to be verified by low temperature experimental observations. Stress-strain curves and dislocation density information are also generated in the simulation [37].

2.2 Nanoindentation Simulation

Verdier, Fivel and Groma have also modeled dislocation dynamics at the mesoscale [18]. Their work presented in "Mesoscopic Scale Simulation of Dislocation Dynamics in FCC Metals: Principles and Applications" (1999) details the effect of image forces on the model, and the use of the finite element technique to simulate the indentation test.

In this study, the basic geometry considered is similar to that which was used in this thesis. The unit cell consists of a cube. The lattice length is assumed to be 26 Å. This is based on the minimum distance in which two dislocations can exist in Cu crystals. Because the crystal type under consideration is FCC, the (110) Burgers vector was used. Glissile dislocations were modeled as straight lines of either pure screw or pure edge character. Screw dislocation segments were allowed to glide along (111) planes in the <110> and <211> directions. Edge segments with line vectors along the <211> direction were allowed to only glide in the <110> direction. This model was taken to be at room temperature and thus climb motion was not included in the scenario.

The dislocation motion is determined through a kinetics scheme which is dependent on the local effective stress. The contributing factors to this stress are the Peierls stress, the elastic stress field generated by the dislocation segments themselves, the line tension of the dislocation, the applied stress, and the image stress generated by the free surfaces.

The velocity of the dislocation segments is determined from the following Newtonian relation

$$v = \tau^{*} \frac{b}{B} \qquad \text{(eq.2.1)}$$

where τ^{*} is the resolved shear stress, b is the burgers vector, and B is a drag coefficient. The time step used was 10^{-9} seconds. A maximum velocity of 100 m/sec is defined. Core effects are modeled in an attempt to take into account the annihilation and recombination of lines, and the formation of sessile junctions.

To determine the features of the dislocation structure of a bulk single crystal, only the central portion of the crystal is considered. For bulk crystal simulation, the cell size used is 15 μm^3 and the dislocation density is taken as 10^{13} m^{-3}. The dislocations inside a 5 μm radius sphere are taken as the only contributing elements to the plasticity of the entire cell. The intention of this methodology is to limit the effects of favoring any crystallographic direction as the segments cross the cube's boundary. Additionally, image forces are thought to be small in the center of the cell so that their effect can be neglected.

To deal with free surface effects in additional simulations, superposition of the elastic field with the image stress field is utilized. The image stress field is analytically determined based on the Boussinesq operator [45]. Because this analytic approach only holds for planar surfaces, an FEM approach is utilized for the more complex problem of the indentation test simulation.

Using the methodology described above, a bulk crystal deformation, a free surface analysis of image forces, and a nano-indentation simulation were conducted [18]. The bulk crystal simulation initiates with a distribution of segments pinned at their ends. The crystal is pulled in tension at a strain rate of $d\varepsilon/dt = 50\ s^{-1}$. The results indicate that the dislocation density steadily increases. It is concluded that a cellular structure is initiated.

The analysis indicates that image forces significantly affect the volume to a depth of 4 μm. If the magnitude of the image force field and the applied stress field are resolved onto the slip system, the image field has only an effect to a depth of 2 μm.

In the nano-indentation test simulation, the computed applied load from the indenter is matched with an experimental value for the displacement. Prismatic dislocation loops are nucleated under the indenter at the locus of maximum shear stress. To obtain the shear stress threshold, the loops are nucleated until the macroscopic force on the indenter is equal to a measured experimental value from a load displacement curve. When the dislocation strain becomes negligible, the relaxation process is aborted and a load increment is applied. This is a quasi-static process. The resulting simulated microstructure is directly comparable to TEM observations.

In a second paper entitled "A Study of the Submicron Indent-Induced Plastic Deformation" (1999), Fivel and Robertson again tackle the problem of simulation of the submicron induced plastic deformation [19]. Their results are compared to experimental findings of indented Cu single crystals. The nanoindentation test indenter is modeled as a 420 nm sphere which penetrates to a depth of 50 nm. Thin foil TEM samples were examined using a Philips CM-20 microscope. These observations are used for comparison with the simulated results. For the numerical simulation of the indentation, the boundary conditions included both forces and displacements imposed on the finite element model surface.

As experimental observations suggest, prismatic loops were nucleated on the active slip systems. Loops were nucleated based on experimental data for displacement as a function of load for single crystals.

The single crystal is modeled as a 2 μm^3 cube. Using the superposition principle, the loading of the indenter as determined from the Hertz formulation is applied to the surface of the FEM model along with the reversed tractions developed by elastic field of the dislocation distribution. The resulting stress field is applied to the straight dislocation segments. The applied load is step increased when the dislocations reach a stable configuration. After the maximum penetration depth is reached the Hertz load is set to zero. Equilibrium is established with only the boundary conditions applied to the top surface.

For comparison with the model, an indentation test was performed on Cu single crystals along the [001] direction. With a loading rate of 0.08 mN/s, load versus displacement curves demonstrate a slope discontinuity. The author concludes that this discontinuity is linked to the effects of an oxide layer. This layer is proposed to be stiffer than the underlying pure Cu and thus sustains some elastic deformation before breaking, or the nucleation of dislocations. TEM micrographs of indented samples show a disk-like dislocation structure with a radius of 600-800 nm. The density of dislocations decreases sharply from the center to the outside of the disk.

Tadmor, Miller and Phillips present another nanoindentation based numerical simulation in “Nanoindentation and Incipient Plasticity” (1999) [27]. In this paper, the initial stages of plastic deformation in an aluminum thin film due to an indenter are simulated. Different crystallographic orientations and indenter geometries are considered. The relationship between load and indentation depth is established. This simulation make use of the “quasicontinuum” method [46].

The energetics of the model is represented by only a small subset of the entire set of atoms. This reduced information is linked to a finite element mesh. The mesh nodes coincide with the representative atoms. The positions of all of the atoms are not accounted for, but can be obtained through interpolation.

The "embedded atom method" (EAM) is used to calculate representative energies [47]. Using the following relation,

$$E_i = \frac{1}{2}\sum_j \phi(r_{ij}) + U(\rho_i) \qquad (2.2.1)$$

the energy of a representative atom is calculated from the relative positions of its neighboring atoms. Here, r_{ij} is the distance from the representative atom to a neighboring atom, $\phi(r)$ is the potential based upon the core-core repulsion of the atomic nuclei, ρ_i is the electron density and $U(\rho)$ is the energy due to attraction of the nucleus to the local electron density. The energy of the representative atom has two components. One component is based on the deformation gradient tensor at this specific position. The other component is based on lattice statics. The equilibrium configuration is computed by minimizing the energy with respect to position through the use of a Newtonian solver. The displacement field in this model does not vary in the z-direction, and conforms to a generalized plane strain form.

The thin film/indenter model includes a 0.1 μm thick 0.2 wide and infinitely long aluminum thin film on a rigid substrate. The model is depicted in figure 2.2.1. below.

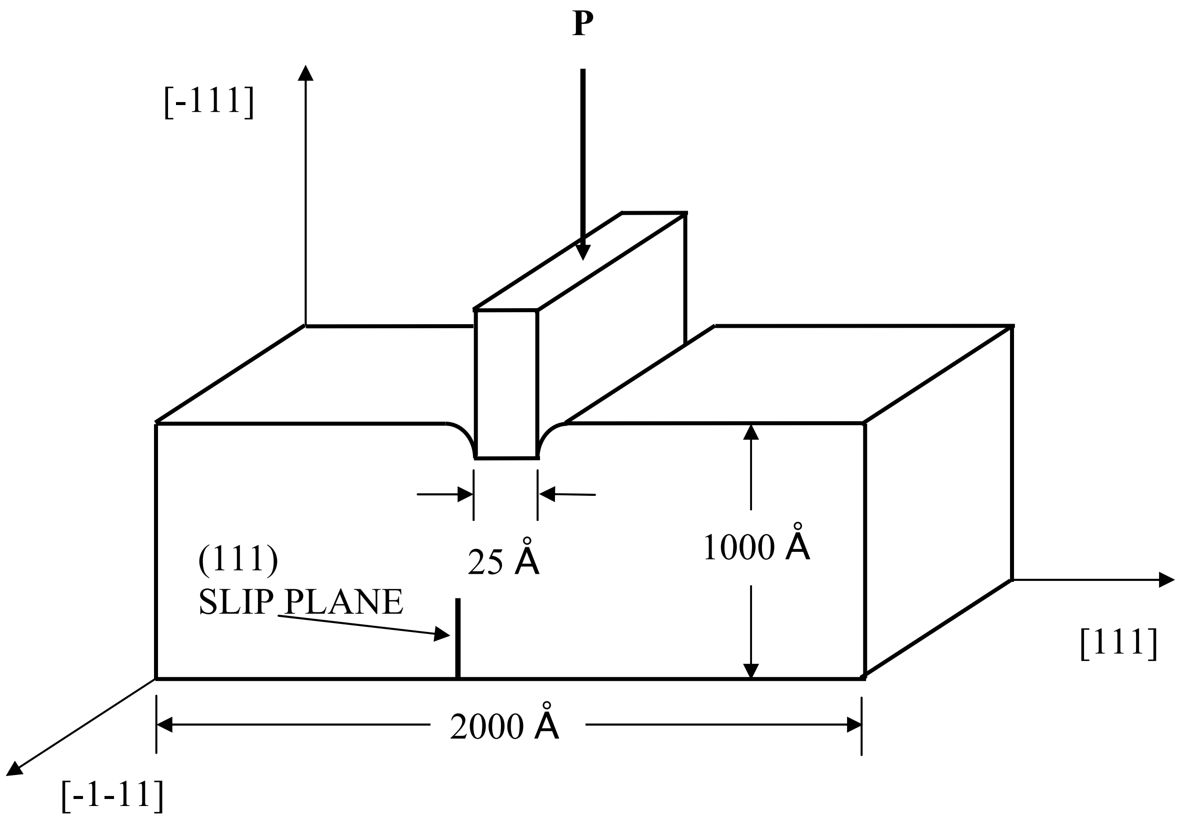


Figure 2.2.1: The indenter/thin film model. (after [27])

Two thin film crystallographic orientations are explored. The first model depicted in figure 2.2.1 above is based on the <110>{111} slip system being parallel to the indentation direction. This scenario is referred to as the “dislocation orientation”. The second model involves the indenter pressed into the (111) surface of the crystal. This model is referred to as the “twinning model”. For both models a rectangular indenter and a cylindrical indenter are explored. The cylindrical indenter has a radius of 11.6 Å.

Indenter-surface interactions are not considered. The top surface is a free surface. This model employed 4,000 representative atoms with 12,000 degrees of freedom. A desktop workstation completed this analysis in a few days.

For the dislocation model, the load-displacement response is linear as predicted by elasticity theory. This linearity ends abruptly causing a load drop due to the emission of edge dislocations beneath the indenter at an indenter depth of 6.7 Å and a load per unit thickness of 24.7 N/m. This corresponds to a mean pressure of 9.8 GPa. This pressure is on the order observed for Berkovich indenters with 0.1 μm tip radii.

The computer simulations [27] indicated that two dislocations are simultaneously emitted. One from the right and one from the left-hand side of the indenter. These dislocations are composed of 1/6 <112> Shockley partials in a staggered, dipole configuration. The dipole settles at a depth of 355 Å. From here the load displacement curve resumes a linear relationship, and then drops suddenly again with the emittance of a second dipole at an indentation depth of 10.8 Å, and a load of 28.3 N/m.

The Peierls stress is estimated based on the equilibrium distance and the embedded atom method. The dipole is assumed to be a single dislocation, which is free to glide in the indentation direction. The equilibrium distance at which all forces balance is sought. The forces at play are Peierls forces, Peach Kohler forces due the indenter, and image forces from the free surfaces. The image forces act to attract the dislocation when it is near the surface and repulse it otherwise. The image force is calculated analytically and the Peach Kohler expression is used for the Peach Kohler force from the indenter stress. The estimated Peierls stress is 8.3 MPa. This is comparable to values predicted in the literature.

A similar series of analyses was undertaken for the cylindrical indenter scheme. Perfect stick conditions characterized contact between the indenter and surface. As the indenter is depressed into the surface, the number of atoms in contact with the indenter increased. After the displacement is completed, any atoms held to the indenter by a tensile load are released and allowed to find an equilibrium position. This caused a stepped surface to develop below the indenter. Because the effective indenter area increases with increasing depth, the load-displacement curve takes a power law form.

A dipole of Shockley partial dislocations is emitted for an indenter depth of 6.6 Å. These partials are the result of the indenter boundary constraints. This occurs at 15.3 N/m and a shear stress of 3.8 GPa. The partials settle at a distance of 42 Å below the indenter. This dipole has a width of 9.5 Å as compared to 23.2 Å for the rectangular indenter. A second partial dislocation dipole is emitted when the indenter reaches a depth of 7Å. The load is 14.7 N/m. The first and second dipole combine to form an edge dipole [27].

Unloading of the system is modeled by pulling the indenter out of the material. Atoms on the surface of the indenter are released when they are held to the indenter by a tensile load. The final configuration consists of a dimple with a depth of one Burgers vector. A dislocation dipole remains in the material unable to overcome the Peierls threshold.

For the twinning scenario with the rectangular indenter, the load displacement curve is linear with small steps associated with the emission of partial dislocations. This is in contrast to the dislocation model which displayed linear regimes punctuated by large vertical drops associated with the emission of edge dislocation dipoles. For the twinning case, the partial dislocation builds slowly as slip develops in the $\left(\bar{1}\,\bar{1}\,1\right)$ plane. This dislocation has a 1/6[112] Burgers vector and a $\left[\bar{1}10\right]$ line direction. Further penetration causes additional dislocations to be nucleated on adjacent planes. A ”needle-like” deformation twinning structure emerges. This structure has been observed experimentally for certain FCC metals like gold under the action of a nanoindenter.

Following small indentations, rosette patterns normally left on the surface of silver disappear. It has been postulated that either the dislocations are annealed out of the material, or that they migrate into the material away from the surface. Unloading the twinning case results in no residual deformation. This has also been experimentally observed.

2.3 Nano and Micro-scale Friction

Hurtado and Kim present a model for single asperity friction slip in "Scale Effects in Friction of Single-Asperity Contacts. I. From Concurrent Slip to Single-Dislocation-Assisted Slip" 1999 [22]. The model used to describe Single-Dislocation-Assisted (SDA) slip consists of spherical particles brought together in adhesive contact by a normal force P. A tangential load T then causes the particles to slide past one another.

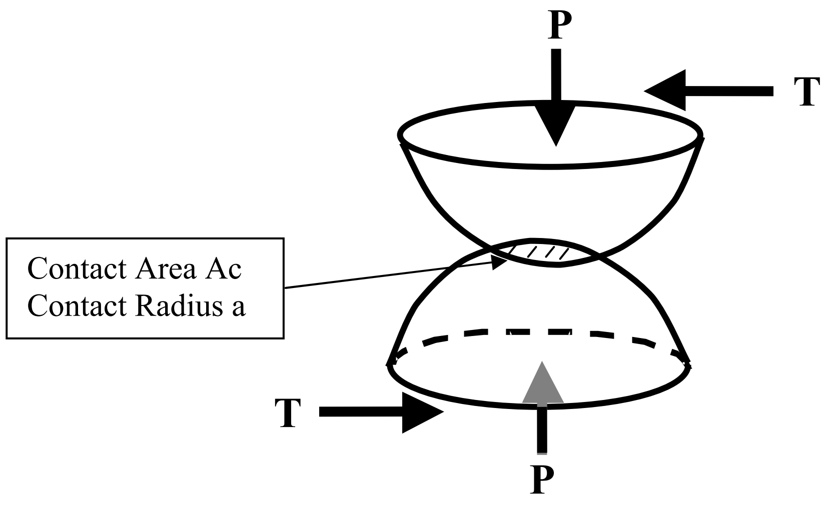


Figure 2.3.1: The Single-Dislocation-Assisted (SDA) friction slip model. (after [22])

The tangential force T is increased to a value of T_f at which slip across the contact area occurs. In the SDA model, a dislocation is nucleated along the perimeter of contact. The shear stress at the contact area causes the dislocation loop to glide in the direction of the Burgers vector, which is in the direction of the shear stress. The dislocation glides and self annihilates. This results in the movement of one asperity over the other. The force on the dislocation consists of the image forces from the surfaces, the self force due to the line tension, the elastic stress field generated by the loop itself, the applied (shear) force, and the Peierls forces from the lattice resistance. The image forces are calculated analytically for a circular dislocation loop concentric with a crack through the use of Green's function. The shear stress required to bring about slip is in good agreement with AFM results. Contact radii over which this model applies are from 10nm to 10 μm. The friction stress is found to decrease with increasing contact radius.

Hurtado and Kim present another model for microfriction in "Scale Effects in Friction of Single-Asperity Contacts. II. Multiple-Dislocation-Cooperated Slip" (1999) [26]. This model applies to contact sizes larger than 10μm in radius. For these dimensions, multiple dislocation loops can be nucleated inside the contact area and begin to pile-up. The model for Multiple-Dislocation-Cooperated (MDC) slip is similar to the model for SDA slip. Nucleated dislocations are concentric, and have the same Burgers vector parallel to the direction of the shear force. Dislocations are assumed to remain circular as they glide through the contact area. Slip of the asperities over one another occurs when the applied shear stress is large enough to cause the central dislocation to overcome the Peierls stress and self annihilate. The stress required for this process to occur is analytically determined.

These two papers describe single asperity slip as consisting of two transitions. For contact sizes between 10nm and 10 μm, the nucleation of a single unstable dislocation loop causes slip. SDA slip is controlled by dislocation nucleation processes. However, for larger contact sizes, the nucleated dislocation is stable. Several dislocation loops may be nucleated and pile-up under the prevailing shear stress. Only once the shear stress is large enough to overcome the Peierls forces will the central dislocation loop glide, self annihilate, and thus allow for the slip of the asperities over one another. MDC slip is a dislocation mobility controlled process. These results are validated by AFM measurements made for shear force versus tip size.

2.4 Experimental Work

Coupeau and Grilhe have used atomic force microscopy (AFM) to observe slip line development during plastic deformation of MC2 and LiF single crystals. Their results are presented in “Quantitative Analysis of Surface Effects of Plastic Deformation" (1999) [17]. This investigation has determined the total number of dislocations emerging at the surface, the average number of dislocations per slip line, and the distribution of terrace widths.

The apparatus used in this investigation consists of a single crystal (2.5 x 2.5 x 5 mm) loaded in compression on the [001] planes. Crystal orientation is close to the <100> direction. The strain rate is 10^{-5} second^{-1}. The atomic force microscope is used to continuously scan the [100] plane during the deformation. The stress strain curve is plotted. The AFM has a scan size of 1 x 1 μm and a resolution of 2 nm. As expected, slip lines emerge from the {111} <110> slip system for MC2, and from the {110} <110> slip system for LiF.

The expected number of dislocations to emerge at the surface is estimated based on the measured value for the macroscopic strain. 25% fewer dislocations emerge than expected. Thus, fewer dislocations are nucleated by the FR sources than expected. These results indicate that the FR sources are locked in the bulk of the material. Slip lines at the surface result from secondary source activation as well as from the movement of dislocations interacting with the FR sources. Thus, macroscopic deformation is composed of slip line structures at the surface, and bulk dislocations within the material. Continuous activation of FR sources is prevented from being the dominant vehicle of plastic deformation.

3 THE METHOD OF DISLOCATION DYNAMICS IN INFINITE ISOTROPIC CRYSTALS

Derivation of the displacement field surrounding a dislocation loop has been solved by N. Ghoniem based on the development by deWit. This derivation along with that of the stress and strain fields, self force distribution, and deformation equations of motion are presented in "Computational Methods For Mesoscopic, Inhomogeneous Plastic Deformation" (1998) [15]. These derivations are summarized below.

3.1 The Displacement Field

A dislocation is formed by making a hypothetical, three dimensional cut through a sold piece of material. The positive side of the cut is held rigid, while the negative side is translated in some direction (see figure 3.1.1).

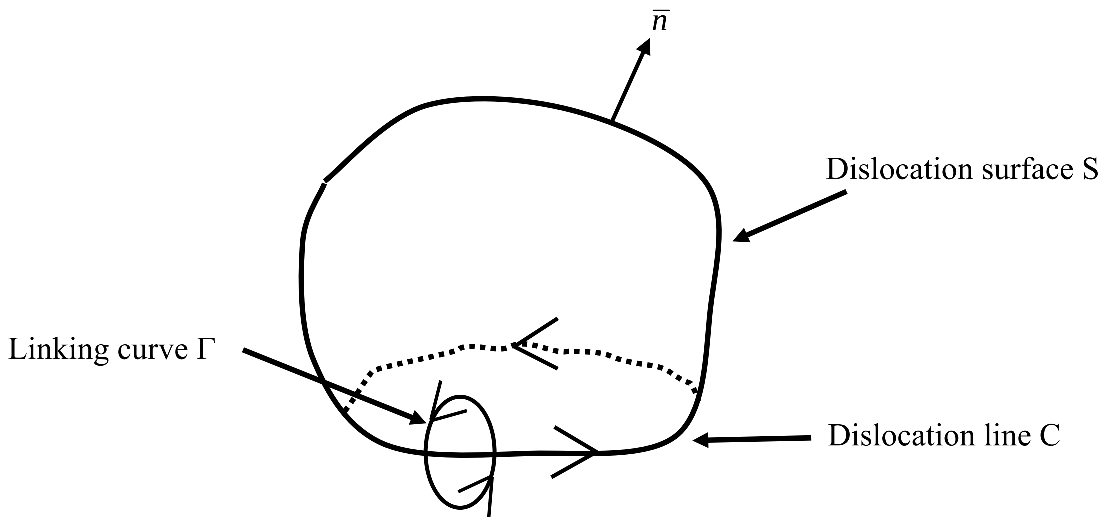


Figure 3.1.1: A cut surface from which a dislocation is created (after [15]).

The Burgers vector **b** is the displacement across the surface of the cut. The dislocation sense vector **t** is the tangent to the dislocation line. Based on a given force distribution $f_m(\hat{\mathbf{r}})$ in the medium under consideration, the displacement vector is expressed by:

$$u_k(\mathbf{r}) = \int_{allspace} U_{km}(\mathbf{r}-\hat{\mathbf{r}})\, f_m(\hat{\mathbf{r}})\, d^3\hat{\mathbf{r}} \tag{3.1.1}$$

where $\hat{\mathbf{r}}$ is the position vector to any point on a dislocation segment, and **r** is the position vector to any point within the medium. $U_{km}(\mathbf{r}-\hat{\mathbf{r}})$ is the isotropic elastic Green's function as given in equation 3.1.2.

$$U_{km}(\mathbf{R}) = \frac{1}{8\pi\mu}\left[\delta_{km} R_{,pp} - \frac{\lambda+\mu}{\lambda+2\mu} R_{,km}\right] \tag{3.1.2}$$

Here, $\mathbf{R} = (\mathbf{r}-\hat{\mathbf{r}})$, $R,_{ij}$ represents repeated differentiation of the radius vector **R** with respect to Cartesian coordinate axes, δ_{ij} is the Kronecker delta, and λ *and* μ are Lame's constants. Through the use of the divergence theorem expression 3.1.1 can be rewritten.

$$u_m(\mathbf{r}) = -b_i \int_{\hat{S}} C_{ijkl}\, U_{km,l}(\mathbf{r}-\hat{\mathbf{r}})\, dS_j \tag{3.1.3}$$

After substitution for the elastic constants tensor (3.1.4) into equation 3.1.3,

$$C_{ijkl} = \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) + \lambda\delta_{ij}\delta_{kl} \tag{3.1.4}$$

the displacement vector can be rewritten as follows:

$$u_m(\mathbf{r}) = \frac{1}{8\pi}\int_{\hat{S}} b_m R_{,ppj}\, d\hat{S}_j + \frac{1}{8\pi}\int_{\hat{S}} (b_l R_{,ppl}\, d\hat{S}_m - b_j R_{,ppm}\, d\hat{S}_j) + \frac{1}{4\pi}\left(\frac{\lambda+\mu}{\lambda+2\mu}\right)\int_{\hat{S}} (b_j R_{,ppm}\, d\hat{S}_j - b_k R_{,kmj}\, d\hat{S}_j) \tag{3.1.5}$$

This equation can be converted to a line integral through Stokes theorem.

$$u_i = -\frac{b_i \Omega}{4\pi} + \frac{1}{8\pi} \oint_C \left[\varepsilon_{ikl} \, b_l \, R_{,pp} + \frac{1}{1-\nu} \varepsilon_{kmn} \, b_n \, R_{,mi} \right] dl_k \tag{3.1.6}$$

This integral can then be converted to a fast numerical sum over quadrature points $(1 \le \alpha \le Q_{\max})$ associated with weighting factors w_α, loop segments $(1 \le \beta \le N_S)$, and number of ensemble loops $(1 \le \gamma \le N_{loop})$. This results in the form used for calculations.

$$u_i = \frac{1}{4\pi} \sum_{\gamma=1}^{Nloop} \left\{ -b_i \Omega + \frac{1}{2} \sum_{\beta=1}^{N_s} \sum_{\alpha=1}^{Q_{\max}} w_\alpha \left(\in_{ikl} b_l \, R_{,pp} + \frac{\in_{kmn} b_n \, R_{,mij}}{1-\nu} \right) \hat{r}_{k,u} \right\} \tag{3.1.7}$$

Where Ω is the solid angle formed by the point of interest with respect to the dislocation line, and $\in_{ijk}$ is the permutation tensor.

3.2 Stress and Strain Fields

The deformation gradient tensor and strain tensors are defined as follows:

$$u_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}) + \frac{1}{2}(u_{i,j} + u_{j,i}) = e_{ij} + w_{ij} \tag{3.2.1}$$

Where w_{ij} is the rotation tensor. Taking the derivative of expression 3.2.1, the deformation gradient and strain tensors are obtained.

$$u_{i,j} = \frac{-b_j \Omega_{,i}}{4\pi} + \frac{1}{8\pi} \oint_C \left[\in_{jkl} b_l \, R_{,ppi} + \frac{1}{1-\upsilon} \in_{kmn} b_n \, R_{,mij} \right] dl_k \tag{3.2.2}$$

$$e_{ij} = -\frac{b_i \Omega_{,j} + b_j \Omega_{,i}}{8\pi} + \frac{1}{8\pi} \oint_C \left[\frac{1}{2} (\in_{jkl} b_l \, R_{,i} + \in_{ikl} b_l \, R_{,j})_{,pp} + \frac{\in_{kmn} b_n \, R_{,mij}}{1-\upsilon} \right] dl_k \tag{3.2.3}$$

Here, ν is Poison's ratio. After substitution for the derivative of the solid angle $\Omega_{,i}$ a fast sum expression can also be derived as.

$$e_{ij} = \frac{1}{8\pi} \sum_{\gamma=1}^{Nloop} \sum_{\beta=1}^{N_S} \sum_{\alpha=1}^{Q\max} w_\alpha \left[-\frac{1}{2} (\in_{jkl} b_i R_{,l} + \in_{ikl} b_j R_{,l} - \in_{ikl} b_l R_{,j} - \in_{jkl} b_l R_{,i})_{,pp} + \frac{\in_{kmn} b_n R_{,mij}}{1-\nu} \right] \hat{r}_{k,u} \tag{3.2.4}$$

The stress tensor is obtained from the strain tensor and converted to a fast sum using the following relations:

$$\sigma_{ij} = \frac{\mu}{4\pi} \sum_{\gamma=1}^{Nloop} \sum_{\beta=1}^{N_S} \sum_{\alpha=1}^{Q\max} b_n w_\alpha \left[\frac{1}{2} R_{,mpp} (\in_{jmn} \hat{r}_{i,u} + \in_{imn} \hat{r}_{j,u}) + \frac{1}{1-\nu} \in_{kmn} (R_{,ijm} - \delta_{ij} R_{,ppm}) \hat{r}_{k,u} \right] \tag{3.2.5}$$

Here, w_α = weighting factors, N_s = loop segments, $Q_{\max}$ = quadrature points, R is the radius vector, Ω is the solid angle between the whole loop and the field point of interest.

3.3 Self Force Calculation

By considering an infinitesimal variation in the position of the dislocation loop over a time interval δt, an expression for the self energy of the loop can be formulated. This formulation as developed by Gavazza and Barnett [14] and presented by Ghoniem [15] is given below as a single line integral over the dislocation loop Γ.

$$\begin{aligned} &\int_\Omega [\sigma_{ik}\, d \in_{ik}]^{(s)}\, d\Omega = Self\ \ Energy = U = \oint_\Gamma f(\kappa,\alpha,\varepsilon) \bullet dr = \\ &\oint_\Gamma \left(\left[E(t) - (E(t) + E''(t)) \ln(\frac{8}{\varepsilon\kappa}) \right] \kappa - J(L,\mathbf{P}) \right) \mathbf{n} \bullet \boldsymbol{\delta}\mathbf{r} \, |ds| + [dU]_{core} \end{aligned} \tag{3.3.1}$$

n is the normal to the slip plane, $\varepsilon = \left|\frac{b}{2}\right|$ is the dislocation core radius, and ϵ is the permutation tensor.

The dominant contribution to the self energy is given by the local curvature κ. This contribution is contained in the energy term E(t) and its second angular derivative E''. The angle between the Burgers vector and the tangent vector is defined as $\alpha = \cos^{-1}\left(\frac{\mathbf{t} \bullet \mathbf{b}}{|b|}\right)$ [15]. Equation 3.3.1 can be rewritten as follows.

$$U = \oint_{\Gamma}\left(-\kappa\left[(E(\alpha) + E''(\varepsilon)\right]\ln(\frac{8}{\varepsilon\kappa}) + \mu b^2\left[\kappa(\frac{21+\cos^2\alpha}{64\pi}) + \bar{\kappa}(\frac{2+\cos^2\alpha - 1}{2\pi})\right]\right)\mathbf{n} \bullet \boldsymbol{\delta}\mathbf{r}\,|ds| \tag{3.3.2}$$

$$E(\alpha) = \frac{\mu b^2}{4\pi(1-\nu)}(1-\nu \quad \cos^2\alpha) \tag{3.3.3}$$

$$E''(\alpha) = \frac{\mu b^2\ \nu}{2\pi(1-\nu)}\cos 2\alpha \tag{3.3.4}$$

$\bar{\kappa}$ is the average curvature of the loop.

The simplification made in rewriting equation 3.3.1 as equation 3.3.2 does not appreciably reduce the accuracy of the self energy calculation [15].

The self force can be thought of as line tension in the dislocation loop. The direction of this force is directed in, towards the center of curvature of the loop. The self force per unit length is found as follows.

$$\frac{\partial \frac{U}{L}}{\partial \mathbf{r}} = \frac{\mathbf{F}}{L} = \left(-\kappa \left[(E(\alpha) + E''(\varepsilon) \right] \ln(\frac{8}{\varepsilon\kappa}) + \mu b^2 \left[\kappa(\frac{21 + \cos^2\alpha}{64\pi}) + \bar{\kappa}(\frac{2 + \cos^2\alpha - 1}{2\pi}) \right] \right) \frac{\mathbf{n}}{|n|} \otimes \xi$$

(3.3.5)

where ξ is the tangent vector to the loop at the point in question.

3.4 Dislocation Deformation

The deformation is formulated based on a variational approach similar to the Galerkin method [15,51]. An integral equation of motion is developed for generalized coordinates. These equations of motion are consistent with the thermodynamics of irreversibility.

Two expressions for the variation in Gibbs energy for the entire loop are obtained through line integration.

$$\delta G^t = -\oint_\Gamma f_k^t \delta r_k |d\mathbf{s}| \leq 0 \qquad 3.4.1$$

Where the total force is the sum of the self force, osmotic force, and the Peach Kohler force ($\mathbf{f}^t = \mathbf{f}_s + \mathbf{f}_o + \mathbf{f}_{PK}$), and δr_k is the displacement of core atoms in the k direction.

$$\delta G^t = -\oint_\Gamma B_{\alpha k} V_\alpha \delta r_k |d\mathbf{s}| \leq 0 \qquad 3.4.2$$

$B_{\alpha k}$ is the resistivity matrix, and V_α is the velocity. These expressions are combined into equation 3.4.3.

$$\delta G^t = -\oint_\Gamma (f_k^t - B_{\alpha k} V_\alpha) \delta r_k |d\mathbf{s}| = 0 \qquad 3.4.3$$

Equation 3.4.3 represents the force balance on every atom of the dislocation.

The dislocation loop is divided into N$_s$ curved segments. Equation 3.4.3 is written in terms of a sum over each segment j.

$$\sum_{j=1}^{N_s}\int_j \delta r_i (f_i^t - B_{ik} V_k)\delta r_k |d\mathbf{s}| = 0 \qquad 3.4.4$$

The dislocation segment is parametrically described through the use of generalized coordinates q$_m$ as follows.

$$r_i = \sum_{m=0}^{N_{DF}} C_{im}\,(u)\,q_m \qquad 3.4.5$$

C$_{im}$ (u) are shape functions dependent on parameter u, where $(0 \le u \le 1)$. N$_{DF}$ is the number of generalized coordinates at the two ends of the dislocation segment. The displacement vector, velocity, and arc length differential for the dislocation segment are given by the following expressions, respectively.

$$\delta r_i = \sum_{m=0}^{N_{DF}} C_{im}\,(u)\,\delta q_m \qquad 3.4.6$$

$$V_k = r_{k,t} = \sum_{n=1}^{N_{DF}} C_{kn}\, q_{n,t} \qquad 3.4.7$$

$$|ds| = (r_{l,u}\, r_{l,u})^{1/2}\, du = \left(\sum_{p,s=1}^{N_{df}} q_p C_{lp,u}\, C_{ls,u}\, q_s\right)^{1/2} du \qquad 3.4.8$$

Equations 3.4.6, 3.4.7, and 3.4.8 are substituted into equation 3.4.4, resulting in the following equation of motion:

$$\sum_{j=1}^{N_s}\int_0^1 \sum_{m=1}^{N_{DF}} \delta q_m\, C_{im}(u)\left[f_i^t - B_{ik}\sum_{n=1}^{N_{DF}} C_{kn}\, q_{n,t}\right]\times\left(\sum_{p,s=1}^{N_{DF}} q_p\, C_{lp,u}\, C_{ls,u}\, q_s\right)^{1/2} du = 0$$

3.4.9

By defining an effective force f_m (equation 3.4.10) and effective resistivity γ_{mn} (equation 3.4.11), expression 3.4.9 can be simplified into expression 3.4.12.

$$f_m = \int_0^1 f_i^t C_{im}(u) \left(\sum_{p,s=1}^{N_{DF}} q_p N_{lp,u} \, N_{ls,u} \, q_s \right)^{1/2} du \qquad 3.4.10$$

$$\gamma_{mn} = \int_0^1 C_{im}(u) \, B_{ik} \, C_{kn}(u) \left(\sum_{p,s=1}^{N_{DF}} q_p \, C_{lp,u} \, C_{ls,u} \, q_s \right)^{1/2} du \qquad 3.4.11$$

$$\sum_{j=1}^{N_s} \left[\sum_{m=1}^{N_{DF}} \delta q_m \left(f_m - \sum_{n=1}^{N_{DF}} \gamma_{mn} \, q_{n,t} \right) \right] = 0 \qquad 3.4.12$$

All local degrees of freedom are mapped into global coordinates. The resistivity matrix γ_{mn} is added into corresponding global locations in the global resistivity matrix Γ_{kl}. This results in the following expression.

$$\sum_{j=1}^{N_s} \sum_{m=1}^{N_{df}} \sum_{n=1}^{N_{DF}} \left[\delta q_m \, \gamma_{mn} \, q_{n,t} \right]^{(j)} = \sum_{k=1}^{N_{tot}} \sum_{l=1}^{N_{tot}} \delta Q_k \, \Gamma_{kl} \, Q_{l,t} \qquad 3.4.13$$

Here, the total number of degrees of freedom for the loop are represented by $N_{tot,}$ ($N_{tot} = N_s \, N_{DF}$). The global force vector F_k is represented as follows:

$$\sum_{j=1}^{N_s} \sum_{m=1}^{N_{DF}} \left[\delta q_m \, f_m \right]^{(j)} = \sum_{k=1}^{N_{tot}} \delta Q_k \, F_k \qquad 3.4.14$$

Thus, equation 3.4.12 can be rewritten:

$$\sum_{k=1}^{N_{tot}} \delta Q_k \left(F_k - \sum_{l=1}^{N_{tot}} \Gamma_{kl} \, Q_{l,t} \right) = 0 \qquad 3.4.15$$

Because the virtual displacements and generalized coordinates are arbitrary, equation 3.4.15 is satisfied as follows.

$$F_k = \sum_{l=1}^{N_{tot}} \Gamma_{kl}\, Q_{l,t} \qquad 3.4.16$$

Equation 3.4.16 is a set of time dependent, ordinary differential equations. These equations describe the motion of the dislocation loops. Equation 3.4.16 can be discretized in time through the generalized trapezoidal family of methods [52]. Through this method, equation 3.4.16 can be rewritten.

$$\sum_{l=1}^{N_{tot}} \Gamma_{kl}^{n+\alpha}\, Q_l^{(n+1)} = \sum_{l=1}^{N_{tot}} \Gamma_{kl}^{n+\alpha}\, Q_l^{(n)} + \Delta t F_k^{(n+\alpha)} \qquad 3.4.17$$

Δt is the time step, n is the time step index, and α is a parameter which determines explicit or implicit integration.

4 PROPOSED FEM/DISLOCATION DYNAMICS APPROACH

The following discussion examines the method by which the force distributions are computed for the single crystal model. Basic numerical procedures are described. Programs and sub-routines that were written to aid in this process are included in appendices.

A dislocation loop is affected by several forces including image forces, self forces, applied forces, Peierls forces, and Peach Kohler Forces. The image forces arise from the interruption of the stress field at the free surface. Self forces are a consequence of the line tension associated with the dislocation loop. Applied forces are induced by the tensile stress applied to the single crystal model. The Peierls forces are a frictional force distribution associated with the resistance of the crystal lattice to the movement of the dislocation. The Peach Kohler forces are forces induced on a dislocation loop due to the elastic stress field of neighboring loops. All of these forces combine to bring about the equilibrium of the loop. Each of these forces will be discussed in the following sections.

4.1 The Peach Kohler Formula and the Forces Affecting The Dislocation Loop

Dislocation loops in this study were defined by several nodal values. Once the stress at these nodal locations was determined, the force per unit length (**f**) on the loop nodes was calculated using the Peach Kohler formula given in equation 4.1.1 [5].

$$f_i = \in_{ijk} \sigma_{jl} b_l \zeta_k \qquad (4.1.1)$$

ζ is the unit vector tangent to the dislocation line, **b** is the Burgers vector, and **ϵ** is the permutation tensor. The total force on the loop is the line integral over the loop circumference.

$$F_i = \oint_s \in_{ijk} \sigma_{jl} b_l ds_k \qquad (4.1.2),[5]$$

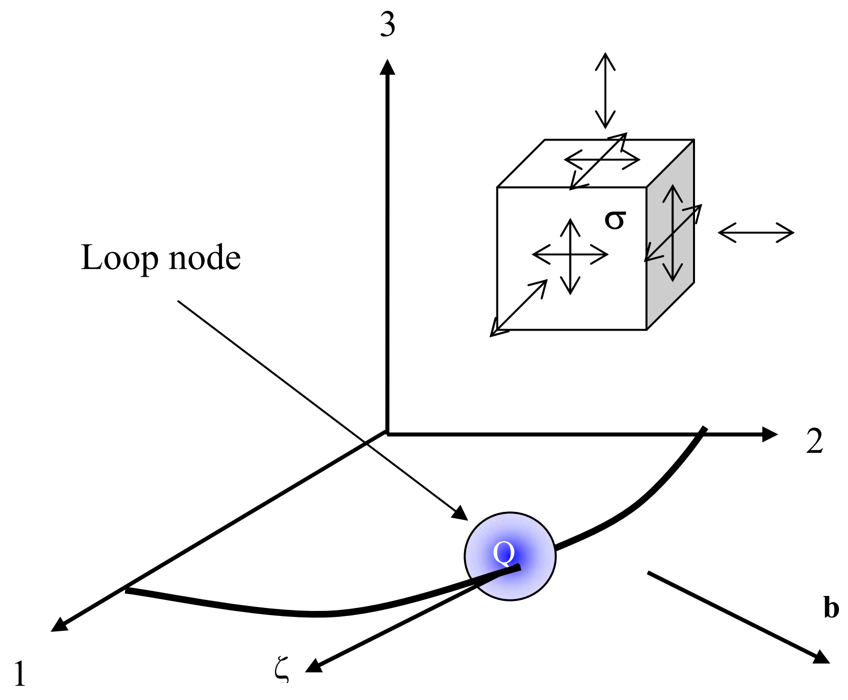


Figure 4.1.1: The dislocation loop and an associated loop node (Q), tangent vector (ζ), Burgers vector (**b**) and stress tensor (**σ**), oriented in 3-D space.

Using equation 4.1.1, the image forces, applied forces, and Peierls forces on the dislocation nodes were calculated using the program "Microplasticity". The self forces were calculated in a different fashion. See Chapter 3 for a discussion of the self force computation.

4.2 Image Forces

Dislocation behavior is profoundly affected by the proximity of the dislocation to surfaces. Depending on the Burgers vector of the dislocation, the surface may be either attractive or repulsive [4]. Dislocation migration toward free surfaces has been observed experimentally [1]. “Pile-ups” occur when dislocations encounter rigid surfaces such as those at grain boundaries. Long range elastic interaction between a dislocation and a boundary take place as a result of image forces. For the case of free surfaces, image forces are the result of the elastic stresses generated by the dislocation itself. The tractions that result from the elastic stresses at a surface must be balanced by opposing tractions so that equilibrium of the surface results.

$$\boldsymbol{\sigma} \bullet \mathbf{n} = \mathbf{t} = 0 \qquad (4.2.1)$$

For traction **t** in equation 4.2.1 to be zero, additional force vectors at the surface need to be introduced to bring about equilibrium. Image forces are the result of the equilibrium condition that must exist at a free surface.

An edge dislocation parallel to a free surface with its slip plane normal to the surface and at a distance l from the surface is depicted in the following figure.

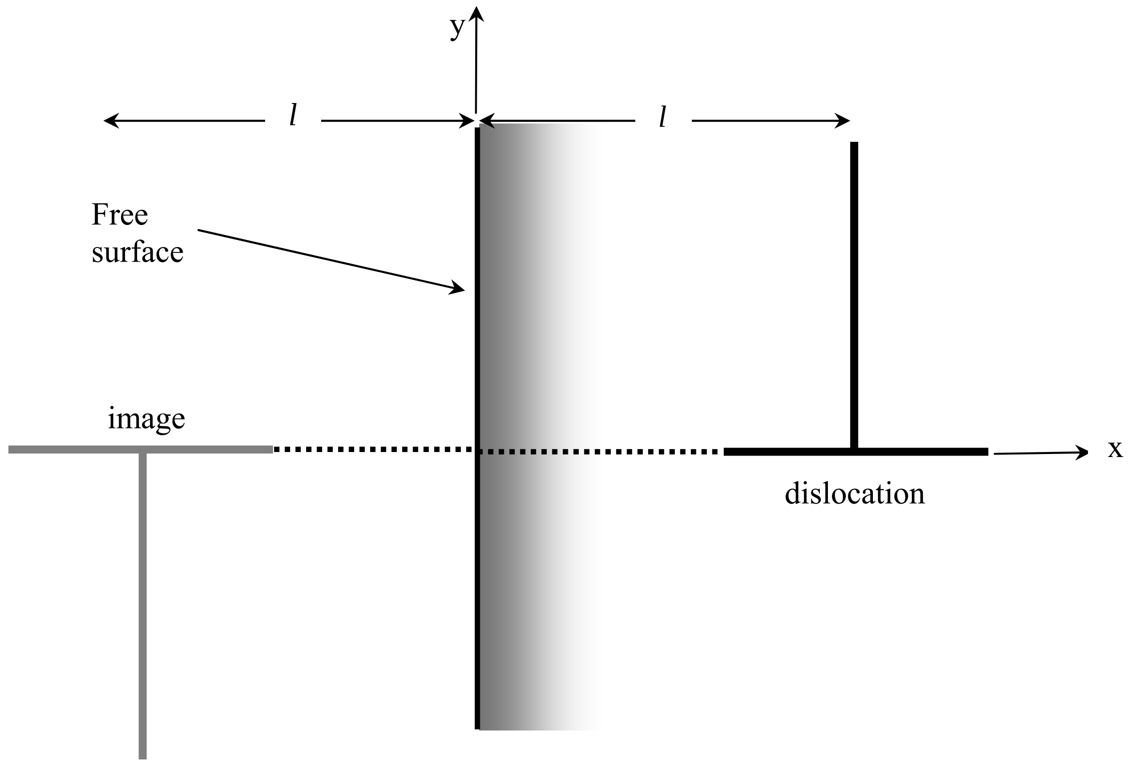


Figure 4.2.1: An edge dislocation and its associated image near a free surface.

Image forces may be thought of as occurring from a hypothetical "image dislocation" on the outside of the surface. For an edge dislocation oriented as shown above, all of the stress components from the image and the dislocation sum to zero at the surface with the exception of the shear stresses (σ_{xy}). This shear stress component at the surface is calculated as follows:

$$\sigma_{xy}(0,y) = \frac{\mu b}{\pi(1-\nu)} \quad \frac{l(l^2 - y^2)}{r^4} \qquad (4.2.2),[2]$$

Where r $=(l^2 + y^2)^{1/2}$, μ is the shear modulus, b is the Burgers vector, and ν is Poisons ratio. This stress component must then be cancelled by addition with a stress function ψ. The following stress fields result from the solution to this stress function and the superposition of this solution with the elastic field of the edge dislocation [2].

$$\sigma_{xx} = -\frac{2\mu blxy}{\pi(1-\nu)r^6}[3(l-x)^2 - y^2]$$

$$\sigma_{yy} = \frac{\mu bl}{\pi(1-\nu)r^6}[4(l-x)^3 y + 6(l-)^2 xy + 4(l-x)y^3 - 2xy^3]$$

$$\sigma_{xy} = \frac{-\mu bl}{\pi(1-\nu)r^6}[(l-x)^4 + 2x(l-x)^3 - 6xy^2(l-x) - y^4]$$

$$\sigma_{zz} = \frac{4\mu bl\nu}{\pi(1-\nu)r^6}[(l-x)^3 y + (l-x)y^3] \qquad (4.2.3),[2]$$

For the case of screw dislocations, the traction free boundary condition is met by the superposition of the elastic field with the stress field generated by the image dislocation alone.

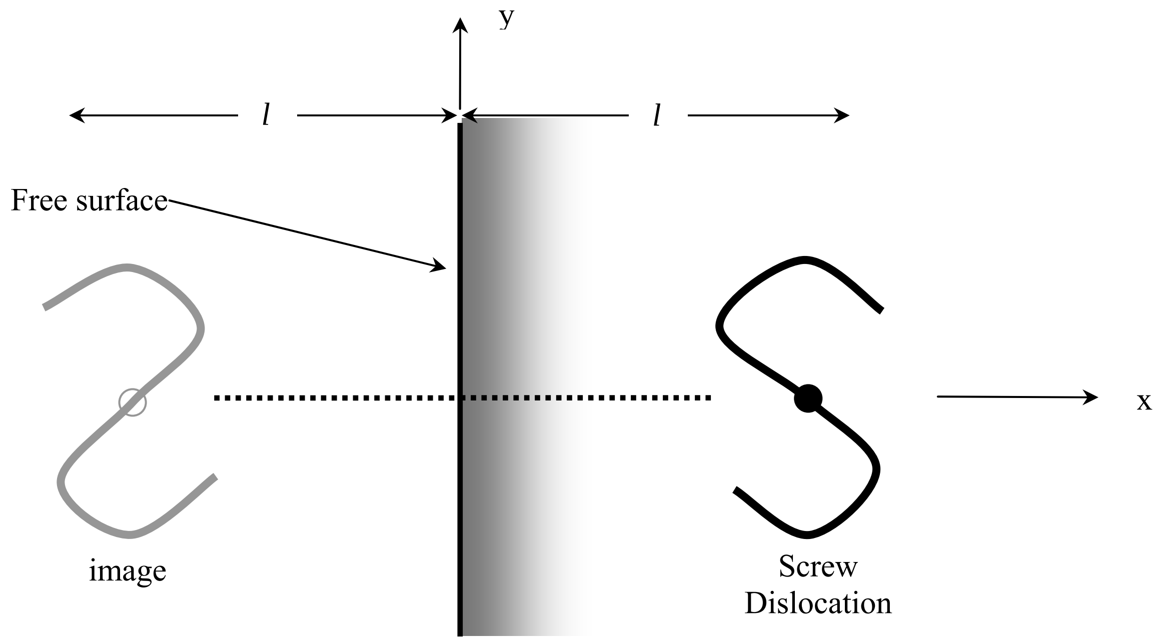


Figure 4.2.2: A screw dislocation and its associated image near a free surface.

All of the stress components developed by the screw dislocation are balanced by the stress components generated by the opposing image dislocation, resulting in no tractions at the free surface.

The effects of image forces on dislocations near free surfaces have been observed experimentally. For very thin Transmission Electron Microscopy (TEM) samples, a "denuded zone" is developed. In this zone near the surface, it is thought that image forces generated by a traction free boundary condition are larger than the Peierls forces, thus causing the dislocations to migrate towards and be removed by the free surface [1].

The image stresses for some simple dislocation-surface interactions can be determined analytically. More complicated interactions such as straight dislocations inclined to a surface and curved dislocations adjacent to a surface, become very complicated and even intractable [2]. For this reason a Finite Element Method was employed in this thesis to calculate image stresses in order to satisfy the free surface boundary conditions required by the single crystal model.

4.3 Finite Element Method for the Calculation of Image Stresses

The image stress field is calculated as depicted in the symbolic diagram below. For the single crystal model, the tensile stress effect on the dislocation loop is calculated through the use of the Peach Kohler formula. To calculate the image stress field, first the elastic stress field induced in the single crystal by the dislocation loop is computed. The tractions that result at the surface of the model from this stress field are then computed, reversed, and placed on the FEM model. The FEM model is then used to calculate the image stress field. From this stress field the Peach Kohler formula can again be used to calculate the image forces on the dislocation loop.

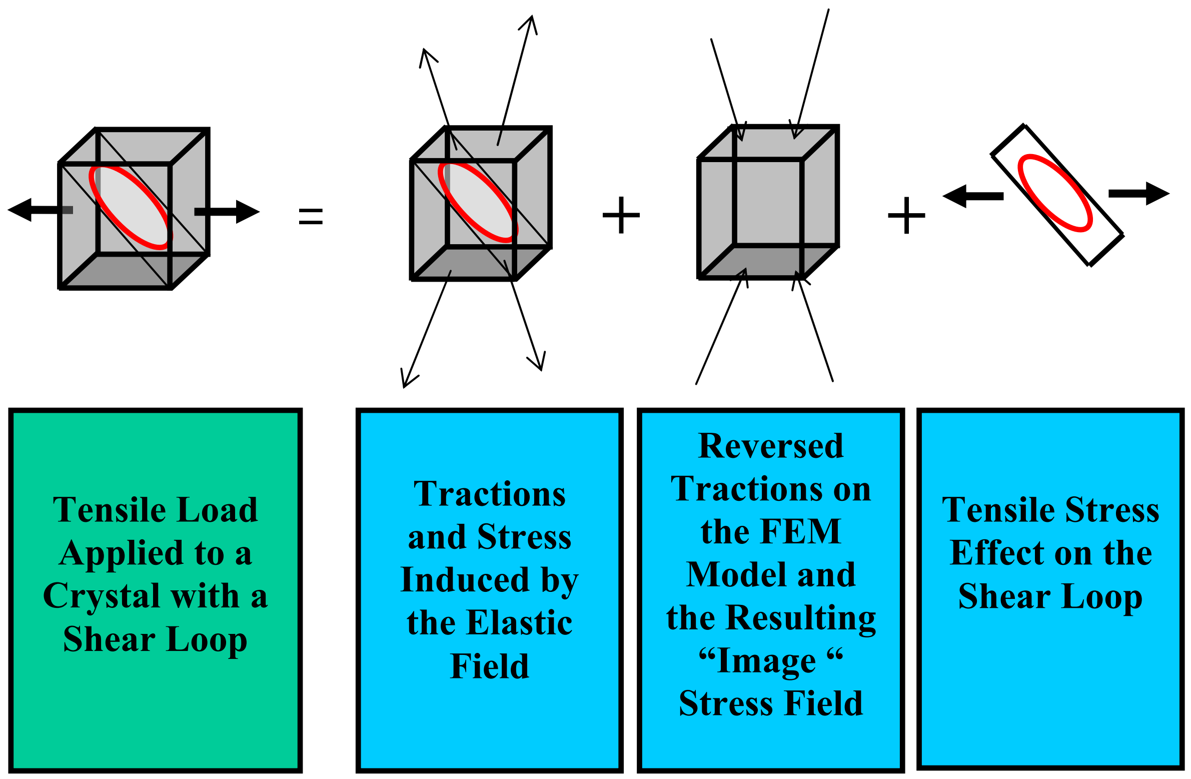

=
+
+
Tensile Load Applied to a Crystal with a Shear Loop
Tractions and Stress Induced by the Elastic Field
Reversed Tractions on the FEM Model and the Resulting "Image " Stress Field
Tensile Stress Effect on the Shear Loop

Figure 4.3.1: Schematic representation of the FEM-Dislocation Dynamics approach used to calculate the stress field in the single crystal model.

Having described the general idea behind FEM-DD approach to the solution of the free surface boundary condition we will now turn our attention to the details of the process. The elastic field developed by the dislocation loop is calculated at the nodal positions defined by the FEM model built in ANSYS and depicted in figure 4.3.2 below. For the bulk of the analyses, a 10 division per side FEM model was employed. This model has 1,331 nodes and 3,990 degrees of freedom. Some analyses were run with a finer mesh generated by using 20 divisions per side. This model resulted in a unit cell with 9,261 nodes and 27,780 degrees of freedom. These FEM models have one node in the center of the crystal model with all degrees of freedom constrained.

Once the ANSYS model is created, the coordinates of the nodes and the associated unit normals are fed into Microplasticity via the input file “Field Input”. The tractions are then calculated using equation 4.2.1 with the unit normal vectors at the surface of the cell as defined figure 4.3.3 below.

The tractions, with their directions reversed, are then imported into the ANSYS model via file “FEM Output”. An analysis is run and the stress field in the interior of the model is computed and written to the file “Ansys Stress”. This file is then manipulated into a format usable by Microplasticity. Programs were written to convert the ANSYS output. These include “Convert”, and “Coordinates and Stress”.

The Peach Kohler formula is then used to calculate the image forces on the loop nodes. Because the FEM analysis only calculates the stress field at specific positions, the stress on the loop nodes has to be estimated by a three-dimensional linear interpolation algorithm [7]. To this end the subroutines “Seeker”, “Loop Node Organizer”, and “Stress Estimator” were written. These subroutines seek the FEM field nodes that are closest to the loop node and estimate the stress on the loop node by using a linear interpolation method described below.

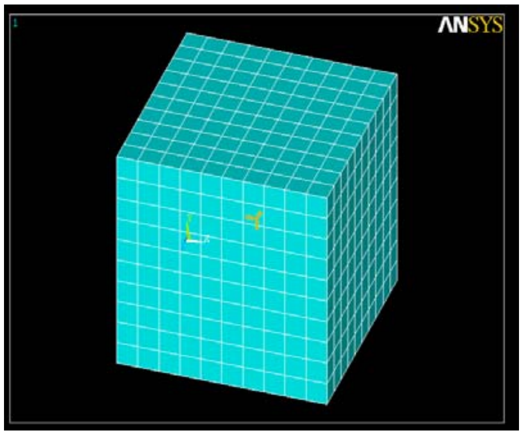


Figure 4.3.2: The FEM model in ANSYS. The model has one central node fully constrained, and 10 divisions per side.

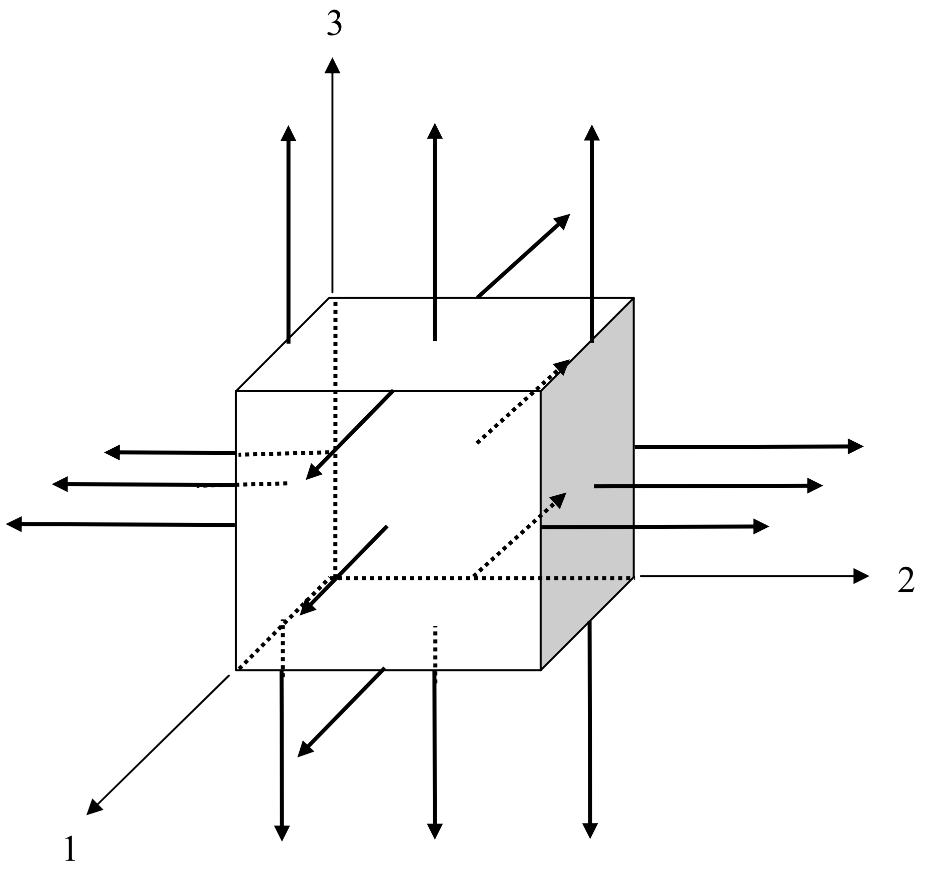


Figure 4.3.3: The model with the unit normal vectors as they were defined for the calculation of the tractions.

4.4 Loop Node Stress Estimator Details

To estimate the stress on the loop node, the stress on the nearest field points surrounding the loop node must be found. These nearest field nodes are referred to as the "capsule". These capsule field node values are found by the subroutine "Seeker". This subroutine compares the FEM field node coordinate values to the loop node in question. The eight field nodes closest to the loop node are selected as the capsule values. The subroutine "Arranger" is then called. This subroutine organizes the capsule stress values and position coordinates so that the "Loop Node Stress Estimator" subroutine can be called upon to then multiply the capsule stress values by the appropriate weighting factors.

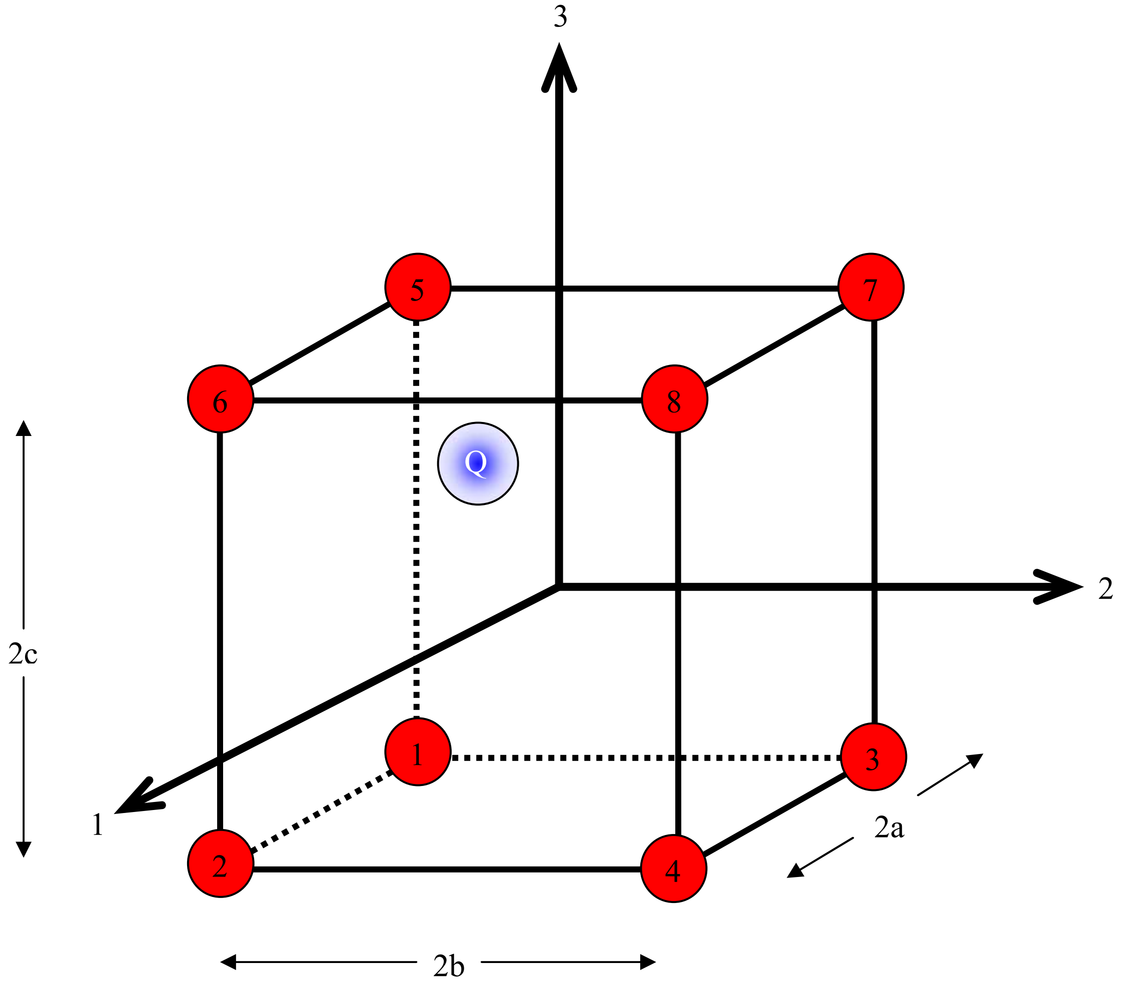


Figure 4.4.1: The loop node Q within the "capsule" of field nodes. Also displayed is the local central origin of the capsule. The length of the sides of the capsule equals the finite element length. These lengths are 2a, 2b and 2c.

The procedure used for estimating the stress on the loop nodes from the stress that was calculated at the field nodes is taken from an extension of the quadratic quadrilateral used in finite element analysis, into three dimensions [7].

Figure 4.4.1 graphically describes the orientation of the field nodes with respect to the loop node. Each of the field nodes corresponds to a specific shape function. The shape functions are given below.

$$N_1 = [(a - x)(b - y)(c - z)] / 8abc \qquad (4.4.1)$$

$$N_2 = [(a + x)(b - y)(c - z)] / 8abc \qquad (4.4.2)$$

$$N_3 = [(a - x)(b + y)(c - z)] / 8abc \qquad (4.4.3)$$

$$N_4 = [(a + x)(b + y)(c - z)] / 8abc \qquad (4.4.4)$$

$$N_5 = [(a - x)(b - y)(c + z)] / 8abc \qquad (4.4.5)$$

$$N_6 = [(a + x)(b - y)(c + z)] / 8abc \qquad (4.4.6)$$

$$N_7 = [(a - x)(b + y)(c + z)] / 8abc \qquad (4.4.7)$$

$$N_8 = [(a + x)(b + y)(c + z)] / 8abc \qquad (4.4.8)$$

In these equations, a, b and c equal half the length of an element in the finite element model. The variables x, y and z are the coordinates of the loop node with respect to the central origin of the capsule. The stress on the loop node is estimated through the following summation:

$$\boldsymbol{\sigma}_{loop\,node} = \sum_{i=1}^{8} N_i\, \sigma_i \tag{4.4.9}$$

here, N_i is the shape factor and σ_i is the stress on the particular field node.

Once the stress on the loop node is established the image force is calculated using equation 4.1.1 as described in section 4.1 above.

The applied stress is calculated in a similar fashion to that of the image force. The applied stress field is entered into Microplasticity and equation 4.1.1 is used to calculate the stress.

4.5 Peierls Forces

The applied resolved shear stress required to overcome the lattice resistance to movement by a dislocation loop is referred to as the Peierls-Nabarro stress. This stress is a consequence of the inter-atomic forces/displacement interaction between the dislocation loop and the surrounding crystal. This resistance to dislocation movement is due to the periodic variation in the misfit energy of atomic half planes above and below the slip plane with the dislocation loop. For high dislocation densities, the influence of the Peierls stress on the dynamics of the dislocation loop is comparable to the long-range interactions between the dislocation loops themselves. For low dislocation densities however, the contribution of the Peierls stress is significant. It is generally accepted that the Peierls stress is a dominant controlling factor in the plastic slip of BCC metals at low temperatures [8, p.246].

Peierls and Nabarro calculated the dislocation energy per unit length as a function of position. This energy was found to oscillate with period b/2, where b is the Burgers vector [1]. This maximum value for the energy is given in the following equation.

$$E_p = \frac{\mu b^2}{\pi(1-\nu)} \exp\left(\frac{-2\pi w}{b}\right) \qquad (4.5.1),[1]$$

The Peierls stress is the critical stress required to move a dislocation through the crystal. This is the maximum slope of the energy vs. distance curve, divide by the Burgers vector.

$$\tau_p = \frac{2\mu}{(1-\nu)} \exp\left(\frac{-2\pi w}{b}\right) \qquad (4.5.2),[1]$$

In equations 4.5.1 and 4.5.2, μ is the shear modulus, b is the Burgers vector, and w is the distance between the atoms immediately below the dislocation.

Values for E_p *and* τ_p are sensitive to interatomic bonding. τ_p varies between

(10^{-6} to $10^{-5}\mu$) for FCC metals, (10^{-2} μ) for covalent crystal like silicon, and is somewhere in between for BCC metals [1]. The Peierls stress decreases with increasing temperature and with increasing dislocation width. It is also lower for edge dislocations than screw dislocations [2]. Based on the Peierls stress, the corresponding Peierls force can be found through the inner product of the Peierls stress with the Burgers vector.

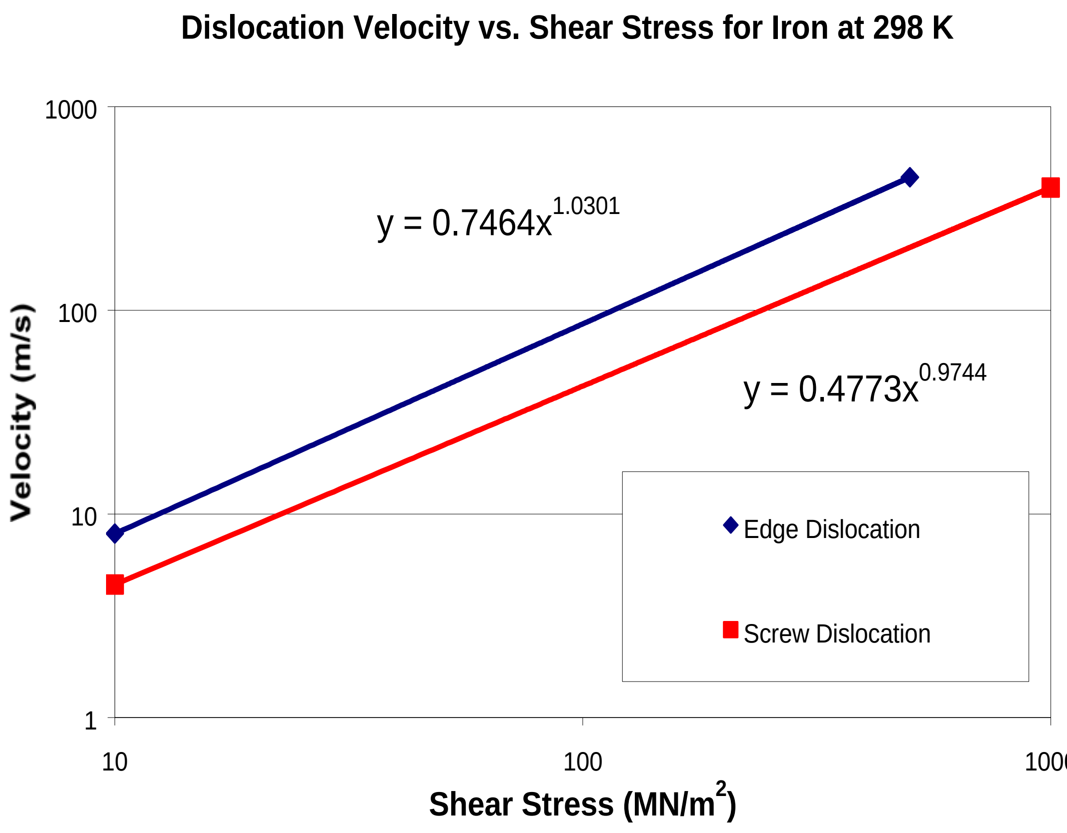


Figure 4.5.1: Comparison of dislocation velocities vs. shear stress for Fe (after[20]).

Based on the graph, the ratio of Fe edge to screw velocity is (2.0). By assuming that the Ratio of Peierls forces associated with edge and screw dislocations are inversely proportional to the velocity ratio, the Peierls force in the direction of the Burgers vector was taken as ½ that in the direction normal to the burgers vector. Other ratios were also explored to determine the variability in the deformed geometry with respect to this variable. The calculation of the Peierls force is given by the following formulation.

$$\frac{\mathbf{F}}{L} = -1 * \tau_p * (1 + \sin(\cos^{-1}\left[\frac{\mathbf{b} \bullet \mathbf{d}}{|\mathbf{b}||\mathbf{d}|}\right]) * \frac{\mathbf{d}}{|\mathbf{d}|} \qquad (4.5.3)$$

The stress threshold value, τ_p is taken as (10^{-3} μ) for Fe, d is the displacement vector, and b is the Burgers vector. The relationship between the Peierls force and the angle between the displacement vector and the Burgers vector is graphically depicted in the figure below.

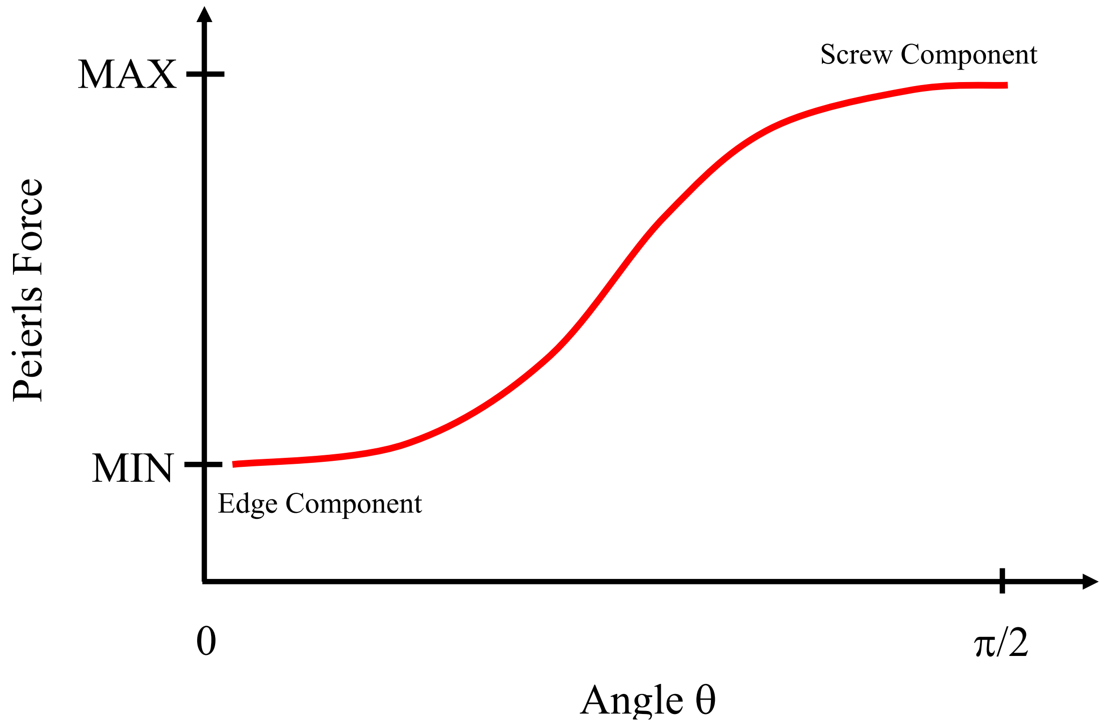


Figure 4.5.2: Sinusoidal relationship between the magnitude of the Peierls Force and the angle between the displacement vector and the Burgers vector.

5 STUDIES OF LOOP FORCE DISTRIBUTIONS AND DEFORMATION NEAR SURFACES IN BCC METALS

5.1 Introduction

Using the finite element method and the method of Dislocation Dynamics described above, the various force distributions on a single shear loop in a BCC Fe single crystal were computed and mapped. The analysis proceeded in the following fashion. First, the displacement, elastic stress and traction fields on the surface of the single crystal model were computed using Microplasticity and plotted using Techplot. Then, stress iso-surfaces for the elastic stress field and the field produced by the superposition of the elastic field and the image stress field were calculated, plotted and compared to verify that the free surface boundary condition was satisfied. To verify the accuracy of the numerical model, all of the forces on the loop in the (101) [-111] slip system were computed and compared to distributions proposed in the literature. Having completed this process, the loop was then placed at different locations within the crystal to study the effects of loop/boundary orientation on the various force distributions. Also, the deformation of the single loop was determined for orientations in close proximity to the boundary. Once the qualitative nature of the force distributions was thoroughly explored, the quantitative relationship between loop/boundary proximity and loop diameter was studied. A final analysis explored the deformation of a FR source for BCC materials with high (10 to 1) and low (2 to 1) ratios of screw to edge Peierls threshold force. The results are discussed below.

5.2 Elastic Field of a Shear Loop in a Finite Crystal

The displacement field for a symmetrically located 4000a-radius shear loop is plotted in figure 5.2.1. The stress distribution for this loop in an infinite medium was calculated. From this stress field the tractions developed at the surface of the crystal were calculated and are plotted in figure 5.2.2A. Based on these tractions, the reversed traction field was imposed on the FEM model of the single crystal as shown in figure 5.2.2B. The image stress field induced by the reversed traction field was computed using ANSYS.

For visual comparison purposes, the traction fields produced by a 2000a-radius loop symmetrically and unsymmetrically located within the slip plane are plotted in figure 5.2.3. The symmetric loop center is located at the local origin of the slip plane. The unsymmetric loop center is shifted by 2000a in both the x and y directions in local slip plane coordinates.

The stress iso-surfaces formed by the σ_{xx} stress component of the elastic stress field and the superimposed stress field formed by adding the image field to the elastic field are then compared in figure 5.2.4. The elastic stress field (in figure 5.2.4A) is observed to cut the surface of the cell in the x = 10000a surface. This results in a traction field at the surface of the single crystal model. The superposition of the image stress field onto the elastic stress field results in a stress field with no normal components at the (x = 10,000a) surface (figure 5.2.4B). The stress iso-surface appears to be repelled from the surface. The normal stress component distribution is negligible at the surface. These results, along with similar results for the shear conponents, show that the free surface boundary condition has been satisfied.

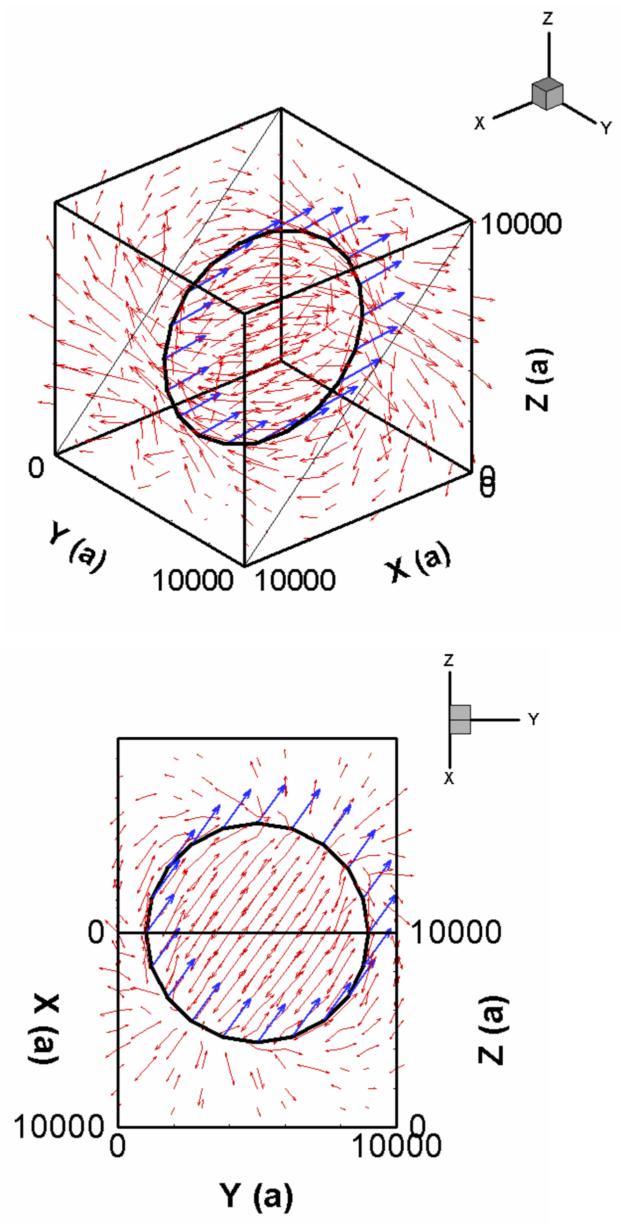
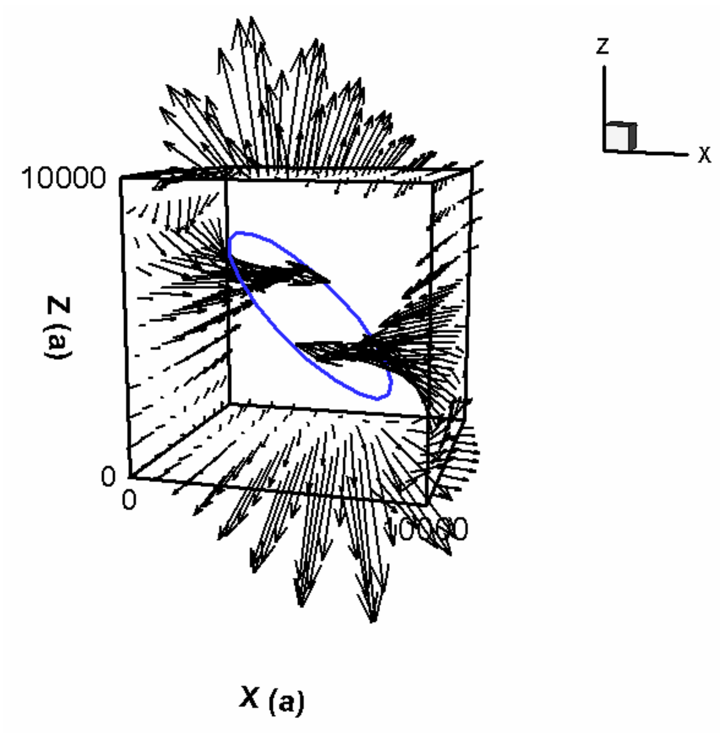
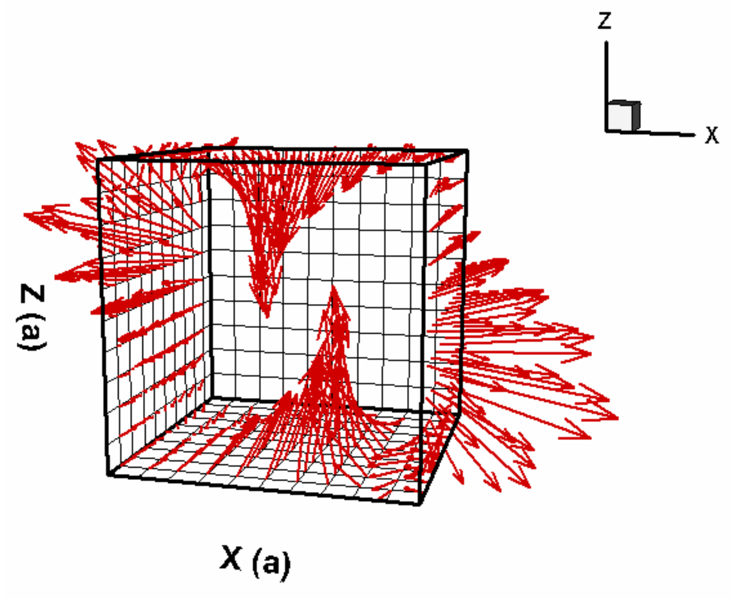


Figure 5.2.1: Two views of the displacement field induced by a 4000a-radius shear loop in the (101)[-111] slip system. Burgers vectors are represented at loop nodes.



(A)



(B)

Figure 5.2.2: Tractions produced by the elastic stress field of a 4000a radius shear loop (A), and the reversed tractions placed on the FEM model of the single crystal (B).

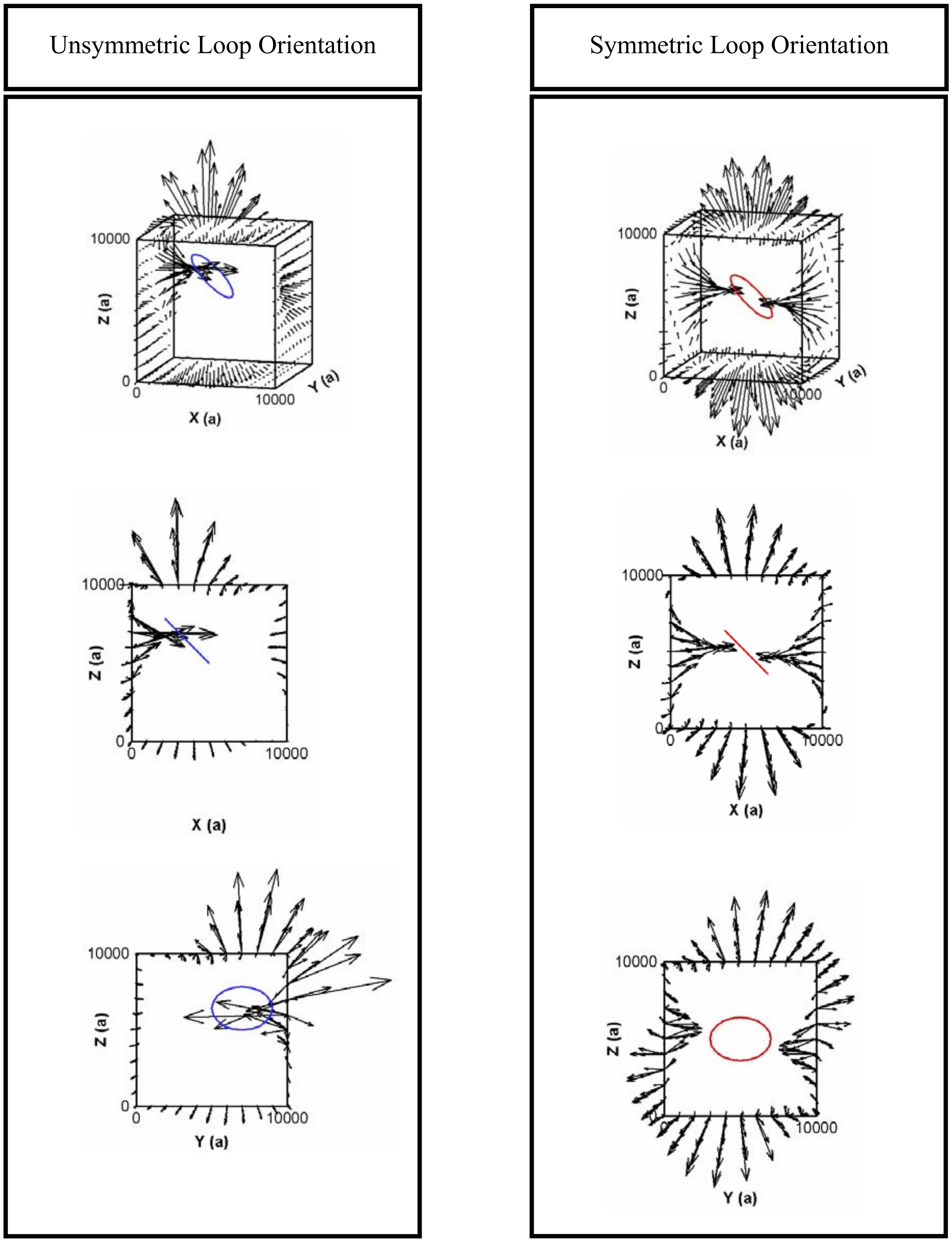
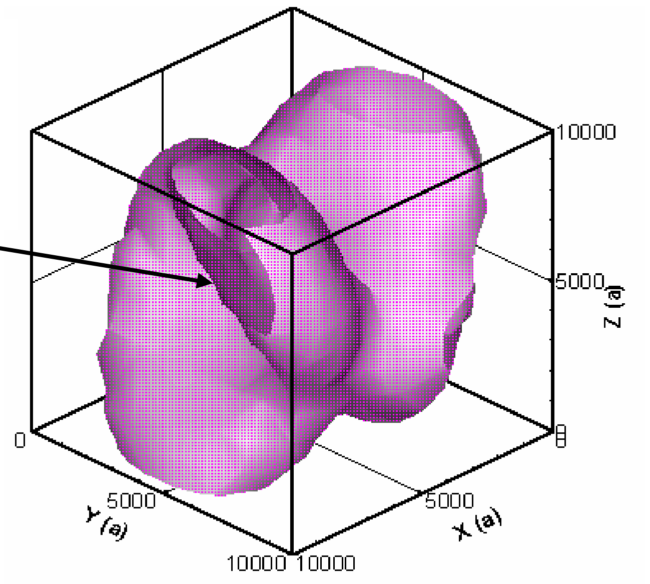


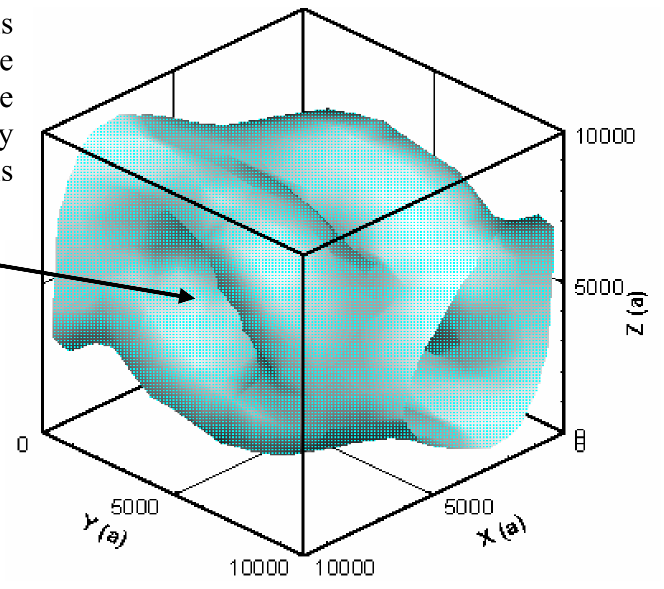
Figure 5.2.3: Three views of a comparison between traction distributions for symmetric and unsymmetric loop orientations (2000a-radius shear loop in the (101)[-111] slip system). Scale factor: 1 (unsymmetric), 2 (symmetric).

***Note:** Stress component pierces the surface. This results in surface tractions.



(A): σ_{xx}=2000 MPa Stress iso-surface; elastic stress field for a loop in an infinite medium

***Note:** Stress component is repelled by the surface. Free surface boundary condition is satisfied



(B): σ_{xx}=2000 MPa Stress iso-surface; superimposition of elastic and image stress fields

Figure 5.2.4: Comparison of elastic stress field before and after superimposition with the image stress field to verify traction free boundary conditions.

5.3 Single Shear Loop Force Distributions

The force distributions on a 4000a-radius shear loop in the (101) [-111] slip system with an applied load of 100 MPa were calculated. The applied, image, Peierls, self and resultant force distributions are plotted below. The image, self and Peierls force distributions are all in the shear plane. The applied and resultant force distributions are characterized by large out-of-plane vector components. The applied force distribution is similar to that proposed by Kroupa [5]. The self force distribution seeks to increase the screw component and decrease the edge component of the dislocation loop. These forces allow for contraction in the direction perpendicular to the Burgers vector. This effect is consistent with the mathematical description of the energy per unit length for edge and screw dislocation components as described by Hull and Bacon [1]. The screw orientation has a lower energy per unit length. Minimization of the edge component and maximization of the screw component thus contributes to lowering the overall energy of the dislocation loop. The Peierls forces are larger in the direction perpendicular to the Burgers vector. These forces allow for expansion in the direction of the Burgers vector. These results are consistent with the idea that edge dislocations are more mobile than screw dislocations [1].

The image force distribution is symmetric about both the vertical and horizontal centerlines of the loop for this specific case. This is due to loop symmetry with respect to the boundaries. The image forces as viewed normal to the slip plane, are larger in the direction of the closer boundary defined by the y planes.

There are negligible resultant forces on some loop nodes. This is because the sum of the image, applied and self forces in the slip plane is less than the Peierls force on these nodes. For nodes were this sum is greater than the Peierls force there is a resultant force vector which includes the Peierls force itself. See Chapter 4 for details on the calculation of forces.

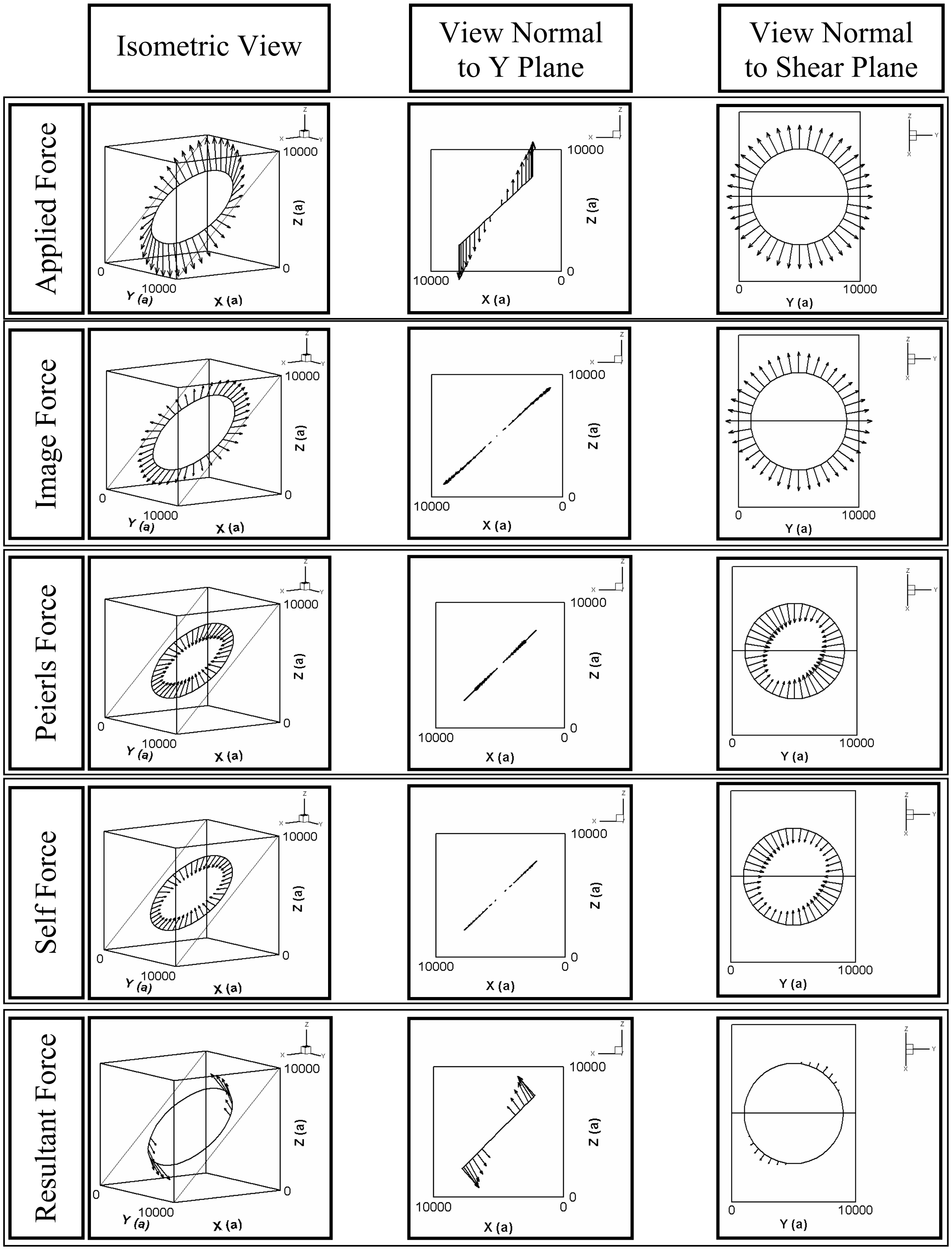


Figure 5.3.1: Force distributions on a 4000a radius shear loop in the (101) [-111] slip system, applied stress σ_{xx} = 100 MPa. Vectors to scale except self and image with scale factors of 2 and 10 respectively.

5.4 Proximity Analyses

Having evaluated and verified the basic force distributions on the single shear loop, and that the free traction boundary condition is satisfied, the qualitative effects of the proximity of the boundary on the image and resultant force distributions was then analyzed. To complete this analysis the loop was placed in different positions within the single crystal model and the force distributions were calculated. In addition to loop position, other variables were evaluated including loop diameter, applied stress magnitude, and number of FEM mesh divisions per side.

For all analyses, the material modeled is BCC iron with a shear modulus of (36.4 GPa), a poisons ratio of (0.25), and lattice constant (a) of (2.85 Å).

5.4.1 Corner Proximity Analyses

The force distribution on a loop with a radius of 2000a initially in the center of the (101) plane was calculated. This loop was then moved in three successive steps toward the corner of the unit cell. Loop center locations in local (101) plane coordinates for each analysis in lattice constants (a) are as follows: (0,0,0), (833,1523,0), (1666,3045,0), (9250,4570,0). The results for each of these analyses are depicted in figures 5.4.1.1 and 5.4.1.2.

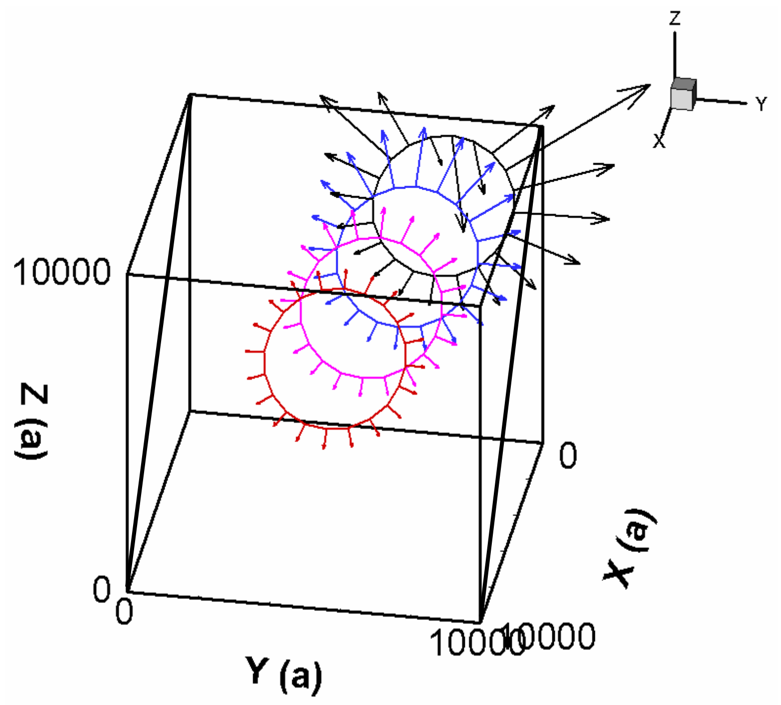


Figure 5.4.1.1: An isometric view of the unit cell with an overlay of image force results from four, non-interacting loops. Loop radius = 2000a.

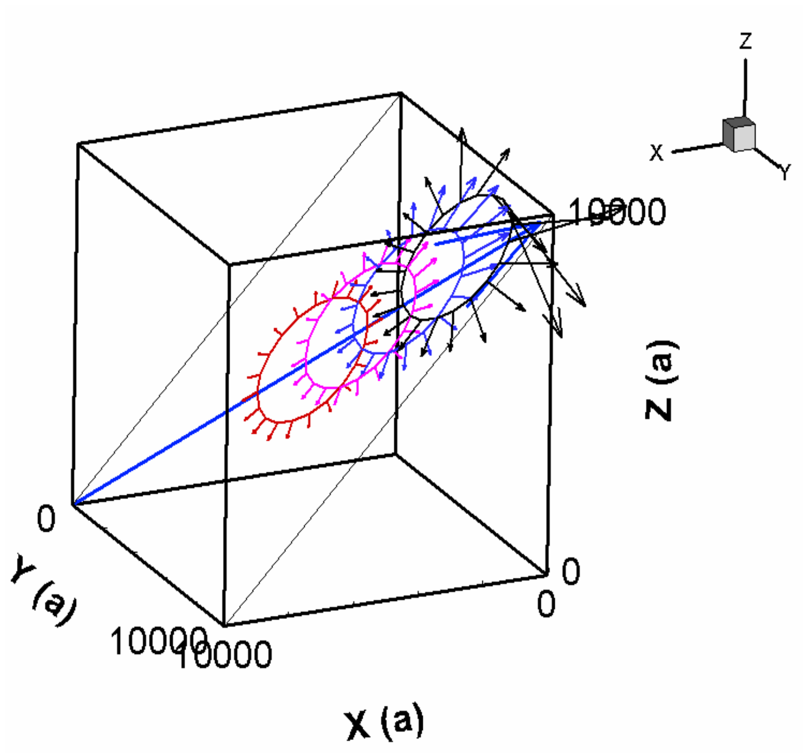


Figure 5.4.1.2: A different view of diagram 5.4.1. The Burgers vector is also represented $[\bar{1}11]$.

Figures 5.4.1.1 and 5.4.1.2 illustrate that as the distance between the loop and the edge decreases, the forces on the portion of the loop closest to the edge are larger than the forces on the portion of the loop far from the free surfaces. All forces are radial and in the slip plane with the exception of the loop that is closest to the corner of the unit cell. Based on these results it would seem that once the loop reaches a critical distance from the edge, the image forces pull the loop out of the slip plane, inducing cross-slip. The distance between the portion of the loop closest to the edge and the unit cell edge itself is 500a.

Similar results were obtained by analyzing the image forces on the dislocation loop as the loop is moved straight up the slip plane toward the edge. These results are displayed in figure 5.4.1.3. This diagram illustrates that as the loop comes close to the edge, an image force distribution develops so as to pull the loop down the side of the unit cell. The minimum distance between the loop and the edge is again 500 lattice constants.

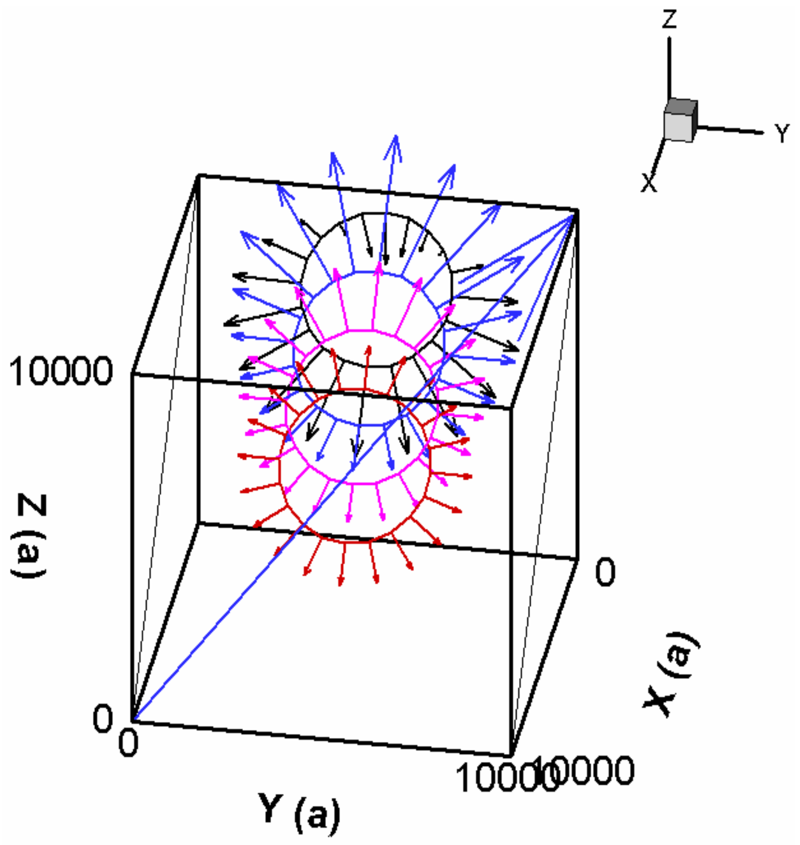
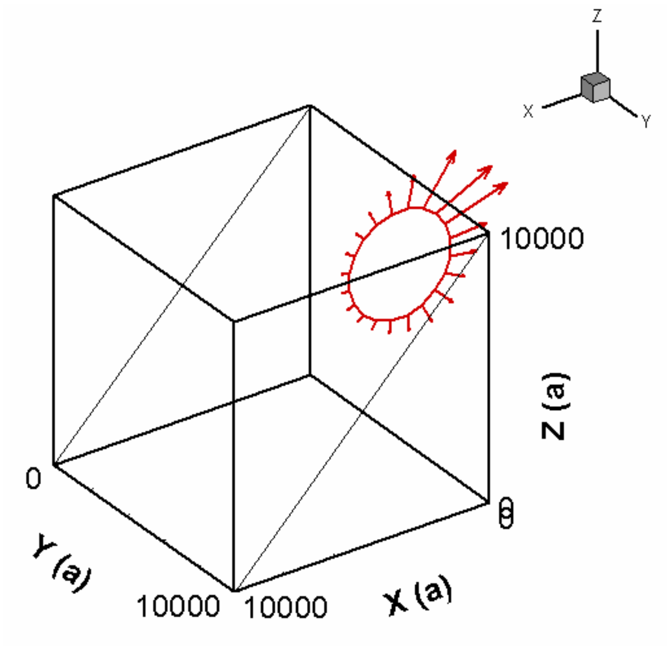
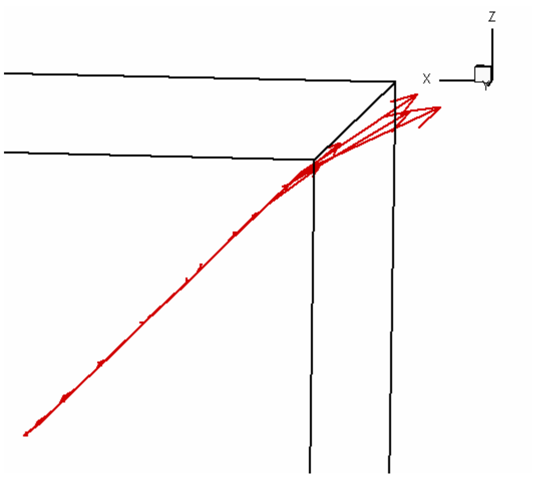


Figure 5.4.1.3: Image force results overlaid from analyses conducted on non-interacting loops placed in different positions in the $(101)[\bar{1}11]$ slip system. The Burgers vector is displayed.

After examining these results two issues surfaced that seemed worthy of additional consideration. Because the FEM model is built with 10 divisions per side, and because the unit normal values are arbitrary at the edges of the unit cell, there may be significant "edge effects" that contribute to the image force field when the loop is placed close to the edge. This computational fault may be responsible for exaggerating the non-planar component of the force distribution on the loop near the unit cell edge. For this reason it seemed prudent to explore the possibility of numerical inaccuracies. To this end, the FEM model was modified such that 20 divisions per side were employed. The location of the loop was maintained at 500 lattice constants from the unit cell edge. The results from this analysis are depicted in the figure below.



(A)



(B)

Figure 5.4.1.4: Two views of the image force distribution on a loop that is 500a from the edge of the unit cell. Analysis conducted with 20 mesh divisions per side.

Figure 5.4.1.4 illustrates that the image forces have a small component off the slip plane. This component is not as significant as that displayed by the less refined mesh analysis depicted in figure 5.4.1.3. The analyses performed with the refined mesh demonstrates that as the loop comes in close contact with the unit cell edge, the FEM element size becomes critical.

To further demonstrate this point, another analysis was conducted with a fine mesh of 20 divisions per side and 60 nodes per loop. The distance between the loop and the edge was decreased to 250 lattice constants. The results of this analysis are illustrated in figure 5.4.1.5 below.

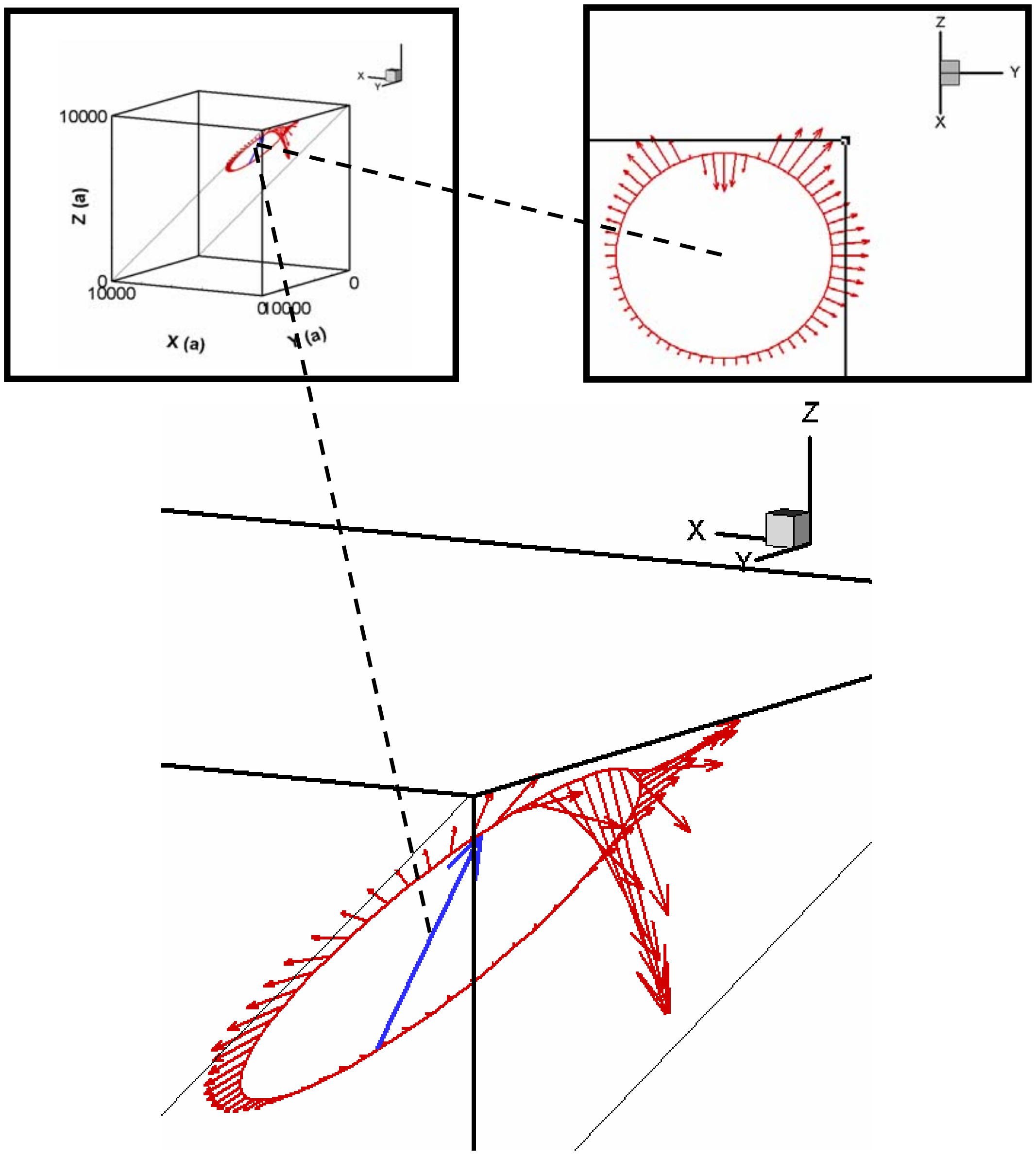


Figure 5.4.1.5: Two views of the image force distribution on a loop in the corner of the unit cell. The loop is 250a from the edge of the cell. These results were obtained using an FEM model with 20 mesh divisions per side. The Burgers vector direction is also represented.

Figure 5.4.1.5. demonstrates that as the loop encroaches on the corner, the image forces distribution increases in magnitude. Image forces on the portion of the loop closest to the boundary are chacterized by repulsive out-of-plane force components. To explore this behavior further, the loop size was reduced to a radius of 500a and the loop was tucked further into the corner.

The results in figure 5.4.1.6 indicate that the image force field is characterized by a distribution that promotes the contraction of the loop. Because loop motion is constrained to the slip plane, image forces effectively repel the loop from encroaching on the free surfaces. Under varying applied stress the resultant force distribution develops so as to cause the loop to elongate along the Burgers vector away from the corner. Even at an applied stress of 350 MPa the loop would not advance on the boundary.

Deformation analysis indicates that the loop advances away from the corner in the direction of the Burgers vector. It must however be noted that the motion of the loop is limited to the slip plane. The repulsion from the boundary is thus an artifact of the numerical modeling limitation. It is obvious from figure 5.4.1.5 that large out-of-plane forces can induce cross-slip.

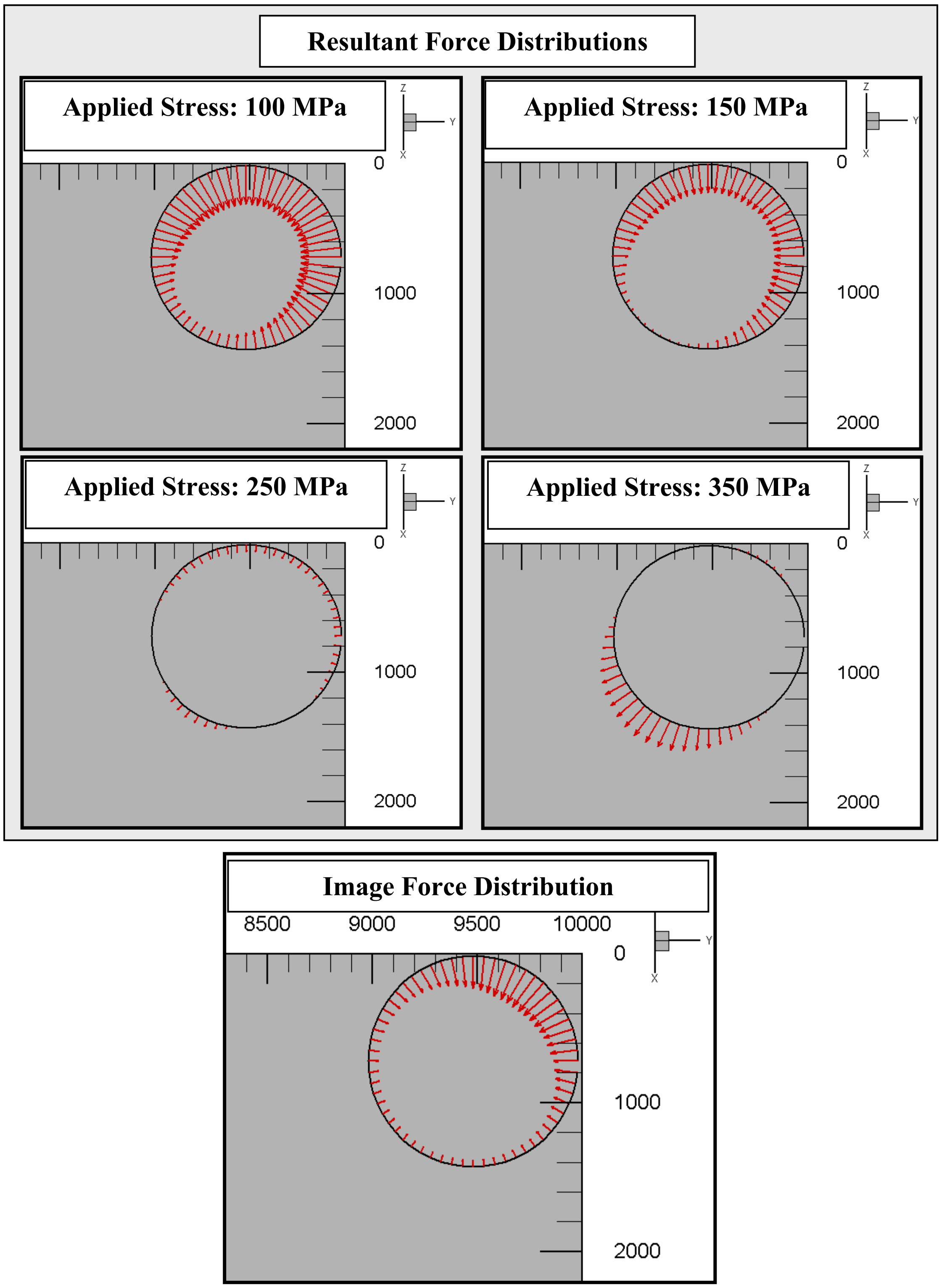


Figure 5.4.1.6: Resultant and image force distributions for a 500a radius shear loop 10a from the edge of the single crystal model for different applied stress levels. All forces to scale.

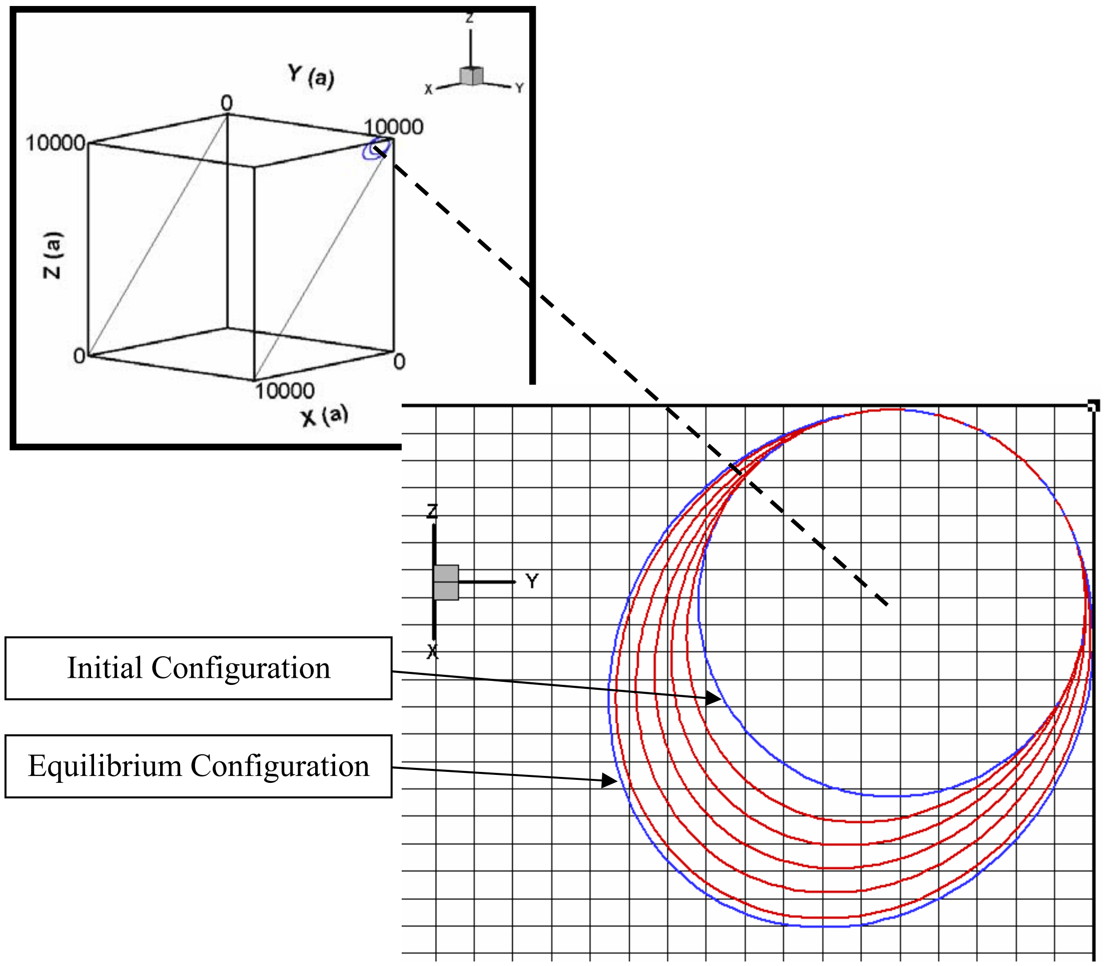


Figure 5.4.1.7: The 500a radius loop inside the crystal structure used in the corner proximity analysis, and the deformation that results under an applied stress of 350 MPa.

5.4.2 Z-Proximity Analyses

To better understand the qualitative relationship between the z = 10000a boundary and the loop in the (101) plane with local coordinates (7500,5000,7500), the following analyses were conducted. A shear loop with a radius of 2000a was placed in different positions with respect to the surface. The image force distributions were plotted.
Figures 5.4.2.1, 5.4.2.2 and 5.4.2.3 depict the image force distributions.

As the loop arrives at the surface, the image force distribution becomes very large. The image forces are radial outward for the loop at the center of its slip plane, 1535a from the boundary (see figure 5.4.2.2). For the loop close to the boundary, this force distribution becomes repulsive when projected onto the slip plane, and is characterized by large off-planar forces on the portion of the loop closest to the boundary. These forces appear to promote cross-slip toward the free surface.

Figure 5.4.2.3 shows that forces on loop nodes not near the boundary evolve from a radial outward distribution at 250a from the boundary, to a downward normal distribution at a distance of 10a from the boundary. These results indicate that the image force distribution is heavily dependent on the proximity to the boundary. To better understand the influence that these boundary effects have on the resultant forces, the resultant force distribution was plotted.

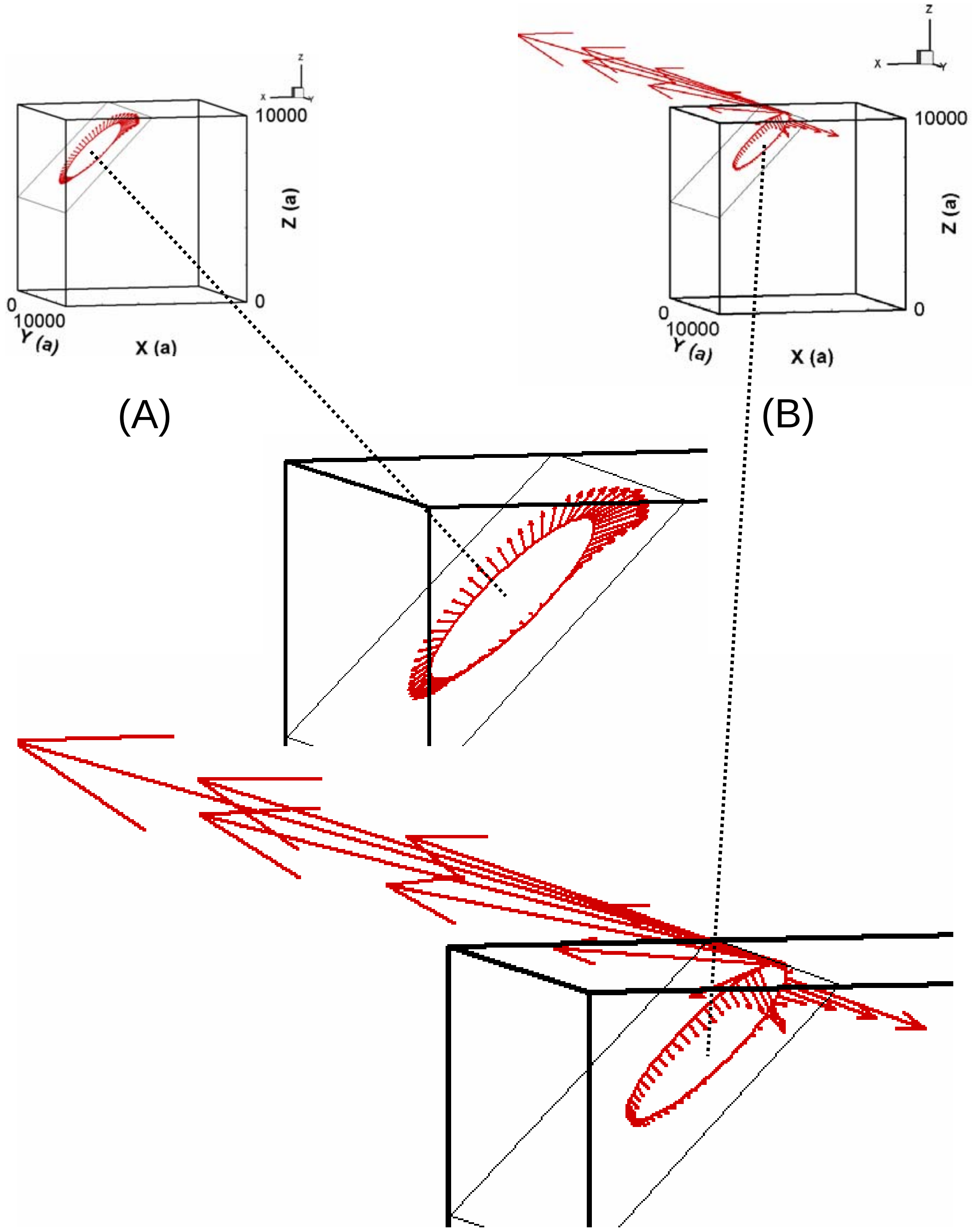


Figure 5.4.2.1: Image force distributions on a 2000a radius shear loop in the (101) [-111] slip system, 1000a (A) and 10a (B) from z=10000a surface. Vector scale factors: 100 (A), 1 (B).

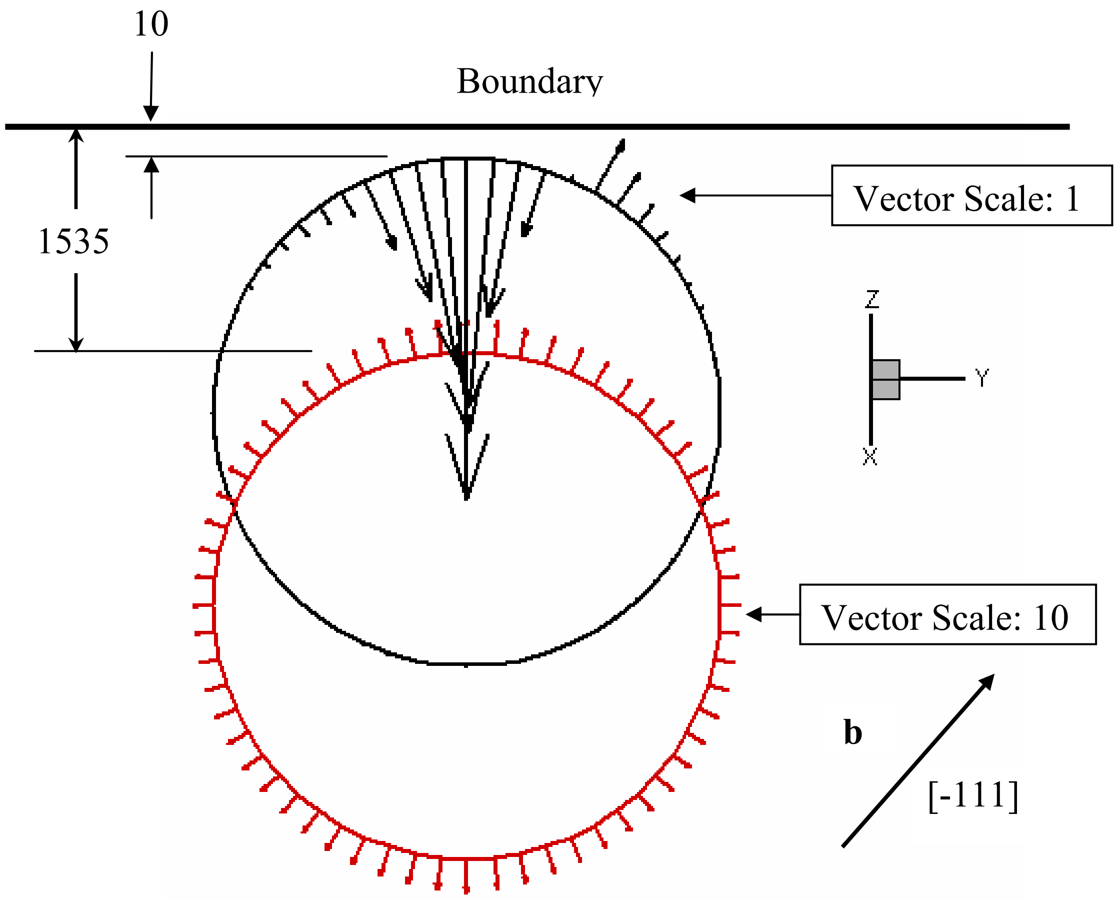
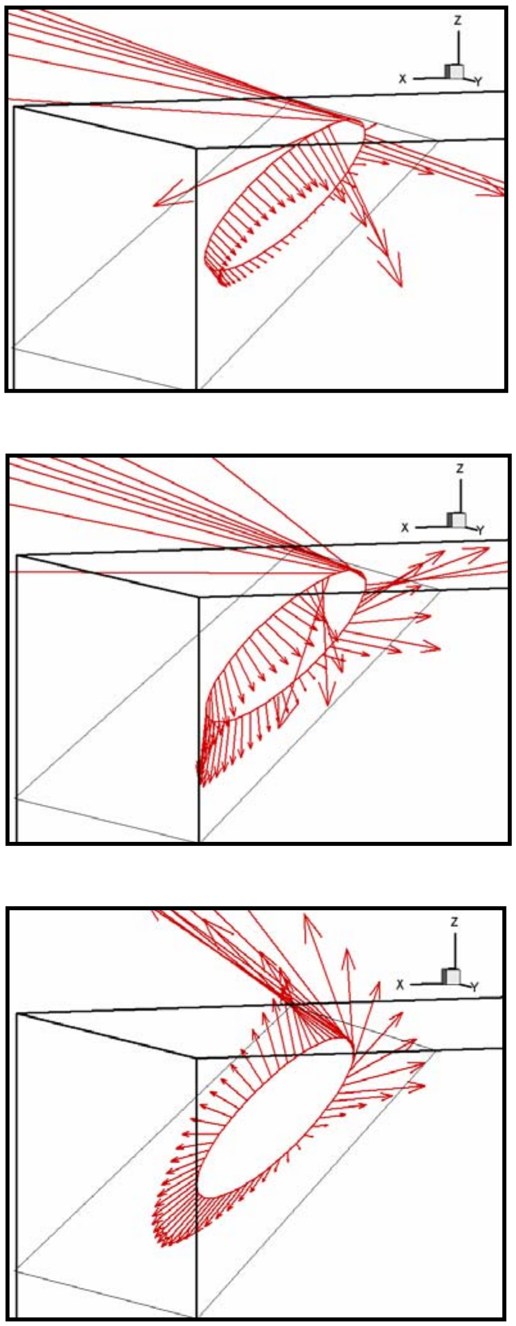


Figure 5.4.2.2: Image force distributions on a 2000a radius loop in the (101) [-111] slip system 10a and 1535a from the z=10000a boundary. Vector scale as indicated.



10a from z = 10000a boundary
scale factor = 1

50a from z = 10000a boundary
scale factor = 10

250a from z = 10000a boundary
scale factor = 10

Figure 5.4.2.3: Image force distributions for a 2000a radius loop at different distances from the z=10000a boundary. Scale factors as indicated.

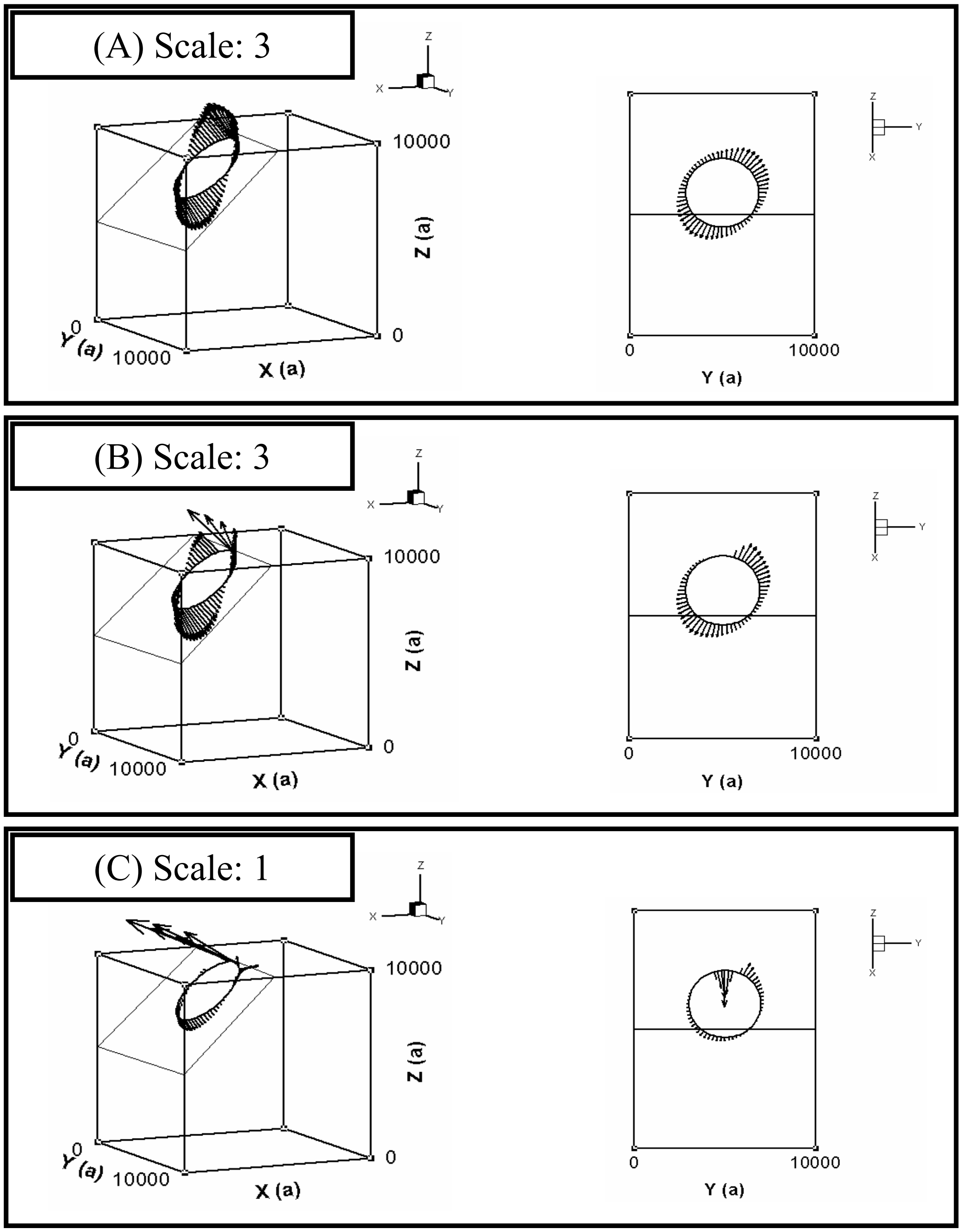


Figure 5.4.2.4: Resultant force distribution on a 2000a radius shear loop 250a (A), 50a (B) and 10a (C) from the z=10000a boundary. σ_{xx} = 200 MPa. Vector scale as indicated.

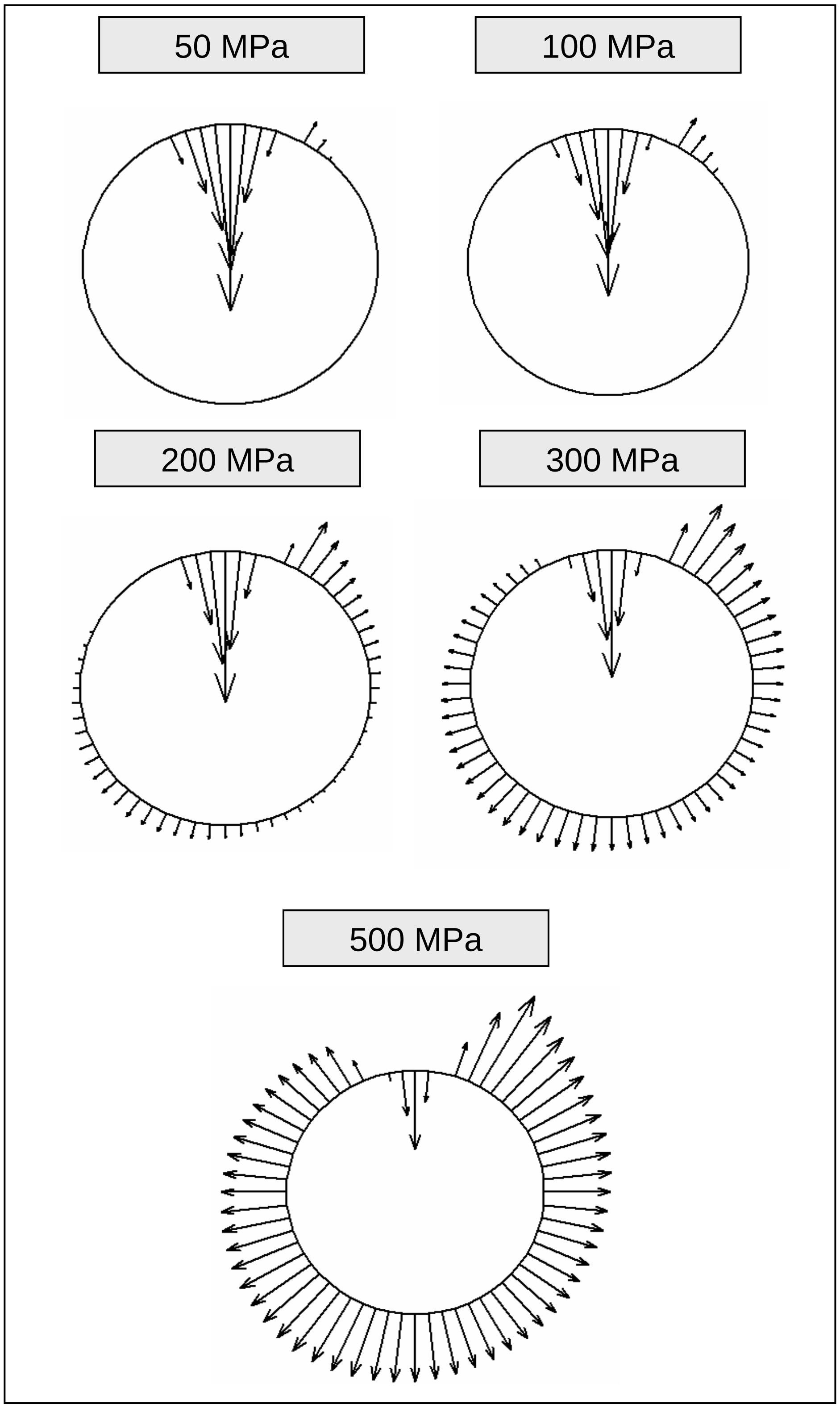


Figure 5.4.2.5: Resultant force distribution on the shear loop 10a from the boundary with varying applied stress. All vector plots are to scale.

Figure 5.4.2.4 depicts the resultant force distributions on the dislocation loop at three different positions, under the influence of a 200 MPa applied stress. As the loop approaches the boundary, the effect of image forces becomes more pronounced. This effect can be seen in the large, repulsive in-plane and large, attractive, out-of-plane forces at the top of the loop at a distance of 10a from the boundary.

Figure 5.4.2.5 depicts the resultant force distribution on the shear loop at a distance of 10a from the boundary with varying applied stress. As the stress is increased from 50 to 500MPa the resultant forces become larger in magnitude with an outward radial character when viewed perpendicular to the shear plane.

Having established the distribution of forces, the deformation program was run to verify the deformed shape of the loop under an applied stress of 50 MPa (see figure 5.4.2.6). As is predicted by the nodal force distribution, the loop is repelled by the surface due to the influence of the image forces, and the in-plane loop motion modeling constraint. A final equilibrium loop geometry is achieved at a distance of 212a from the surface. If the model allowed for motion out of the slip plane, the loop would advance toward the boundary through cross-slip.

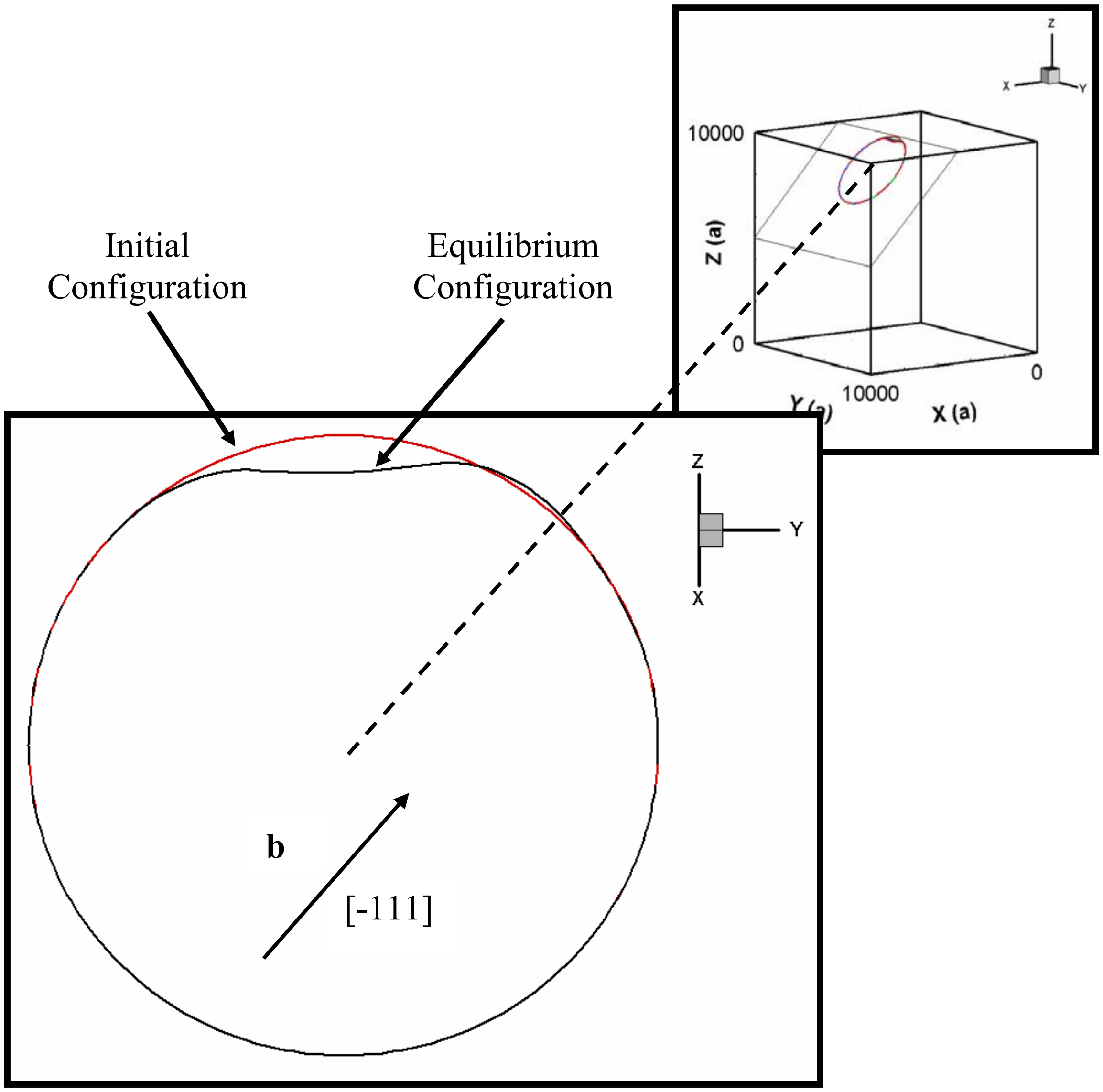


Figure 5.4.2.6: Deformation of a 2000a radius shear loop in the (101) [-111] slip system. The loop is initially 10a from the surface with applied stress: $\sigma_{xx} = 50$ MPa.

5.4.3 Y-Proximity Analyses

A 4000a radius loop was shifted in the (101) plane toward the y=10000a plane. The resultant and image force distributions were plotted. Several locations were analyzed including 10, 100, and 250 lattice constants from the edge of the 10000a single crystal model. An applied tensile stress of 50 MPa was maintained throughout these analyses. Finite element models with 10 and 20 mesh divisions were used to calculate the image stress field.

As the results below indicate, the image forces are larger nearest the boundary. These forces are radial and in the slip plane. The image forces act in such a way as to pull the loop into the boundary when the loop is at the 1000a distance from the edge of the crystal. Once the loop gets very close to the boundary however, this force distribution takes a dramatic turn and causes repulsion from expansion into the boundary.

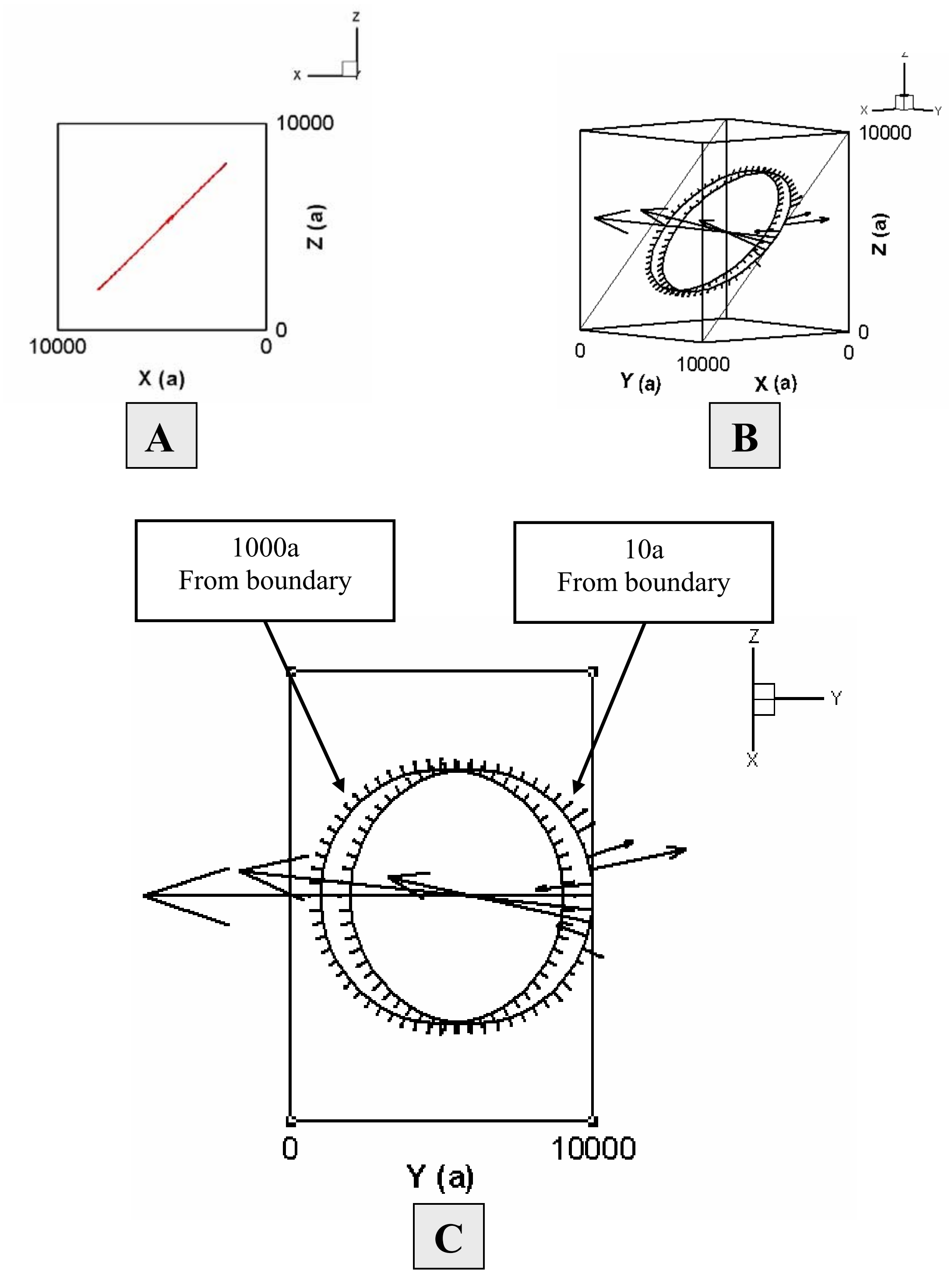
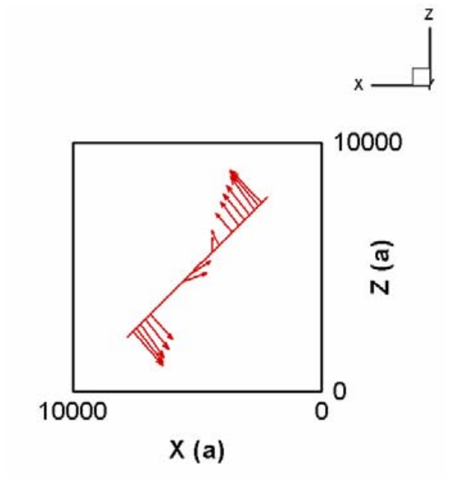
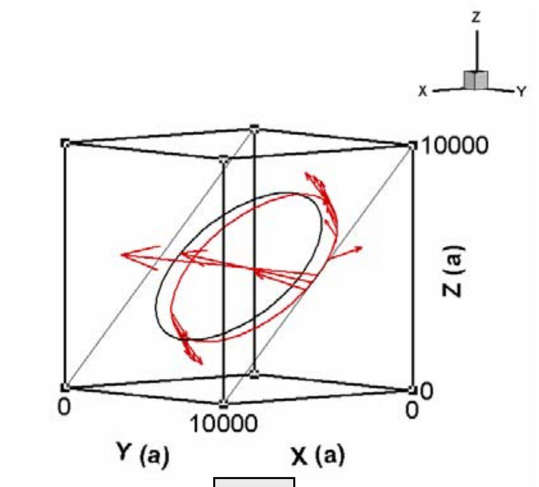


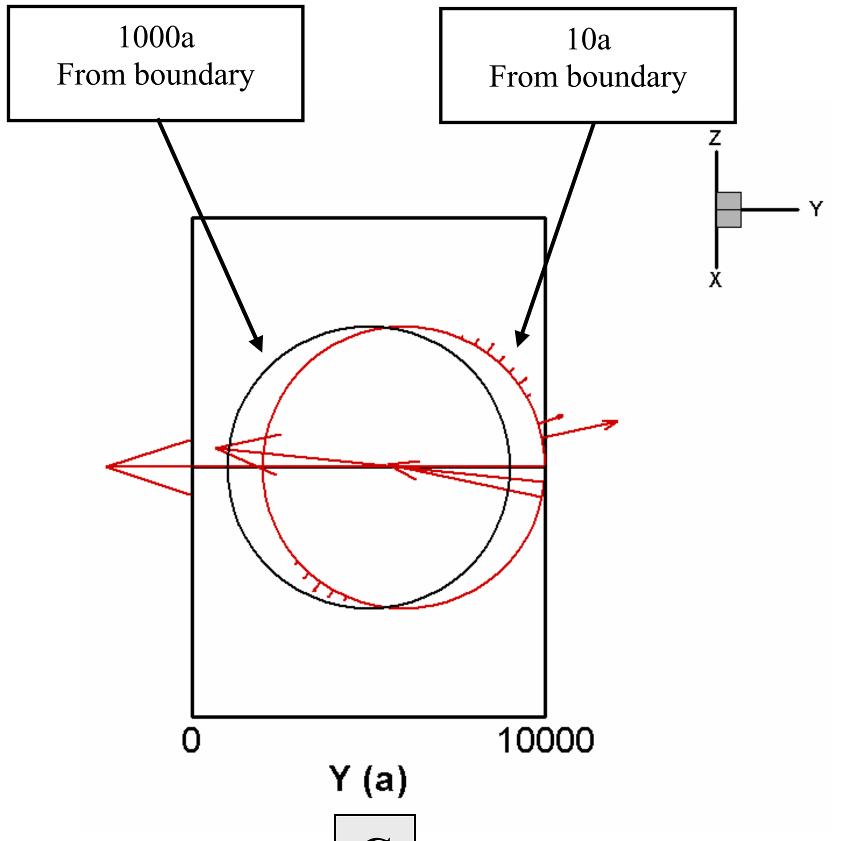
Figure 5.4.3.1: 2-d (A and C), and isometric (B) plots of the image force distributions on 4000a radius shear loops at 1000a and 10a distances from the y=10000a boundary for the (101) [-111] slip system.



A



B



C

Figure 5.4.3.2: 2-d (A and C), and isometric (B) plots of the resultant force distributions on 4000a radius shear loops at 1000a and 10a distances from the y=10000a boundary for the (101) [-111] slip system. σ_{xx} = 50 MPa.

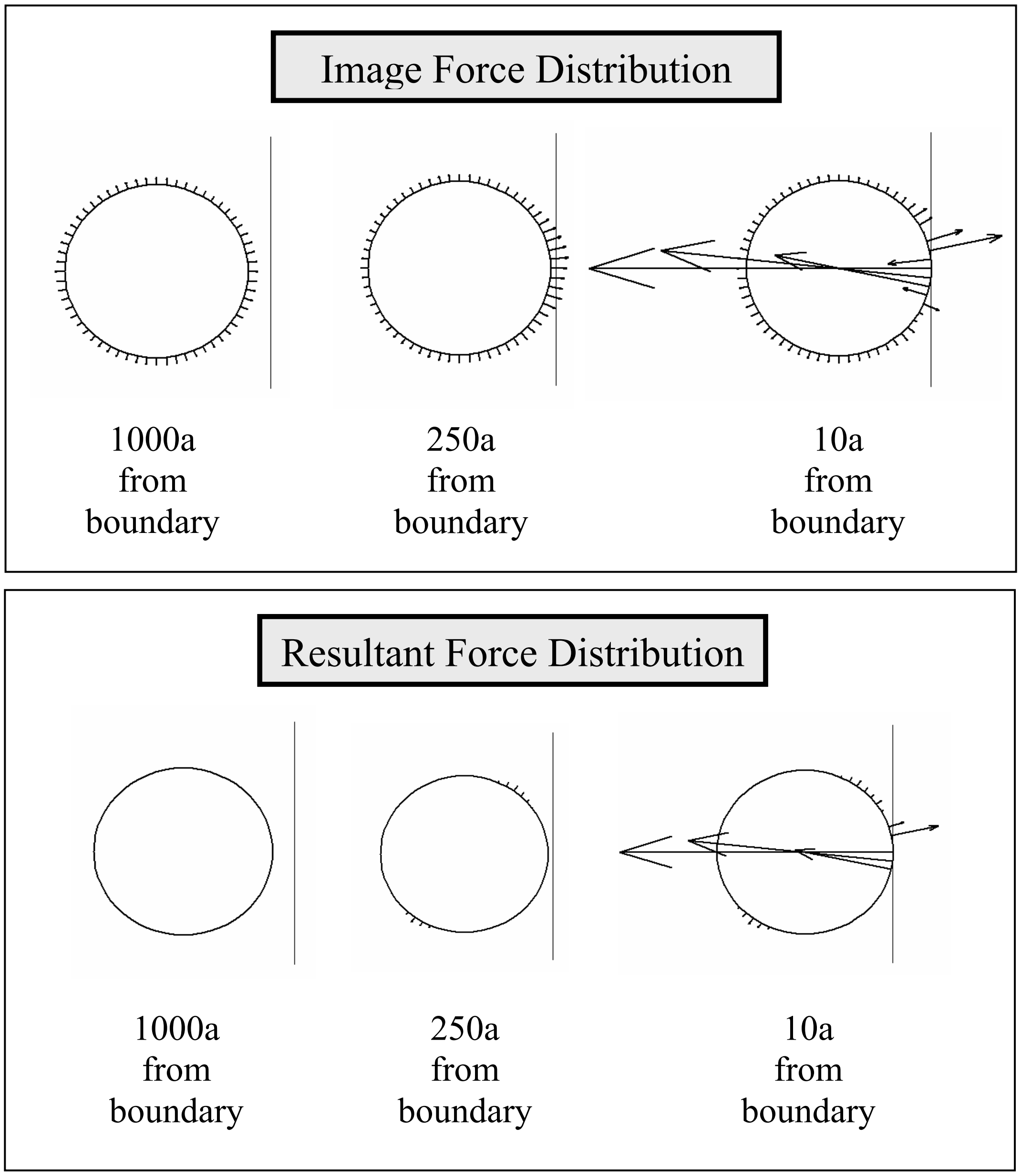
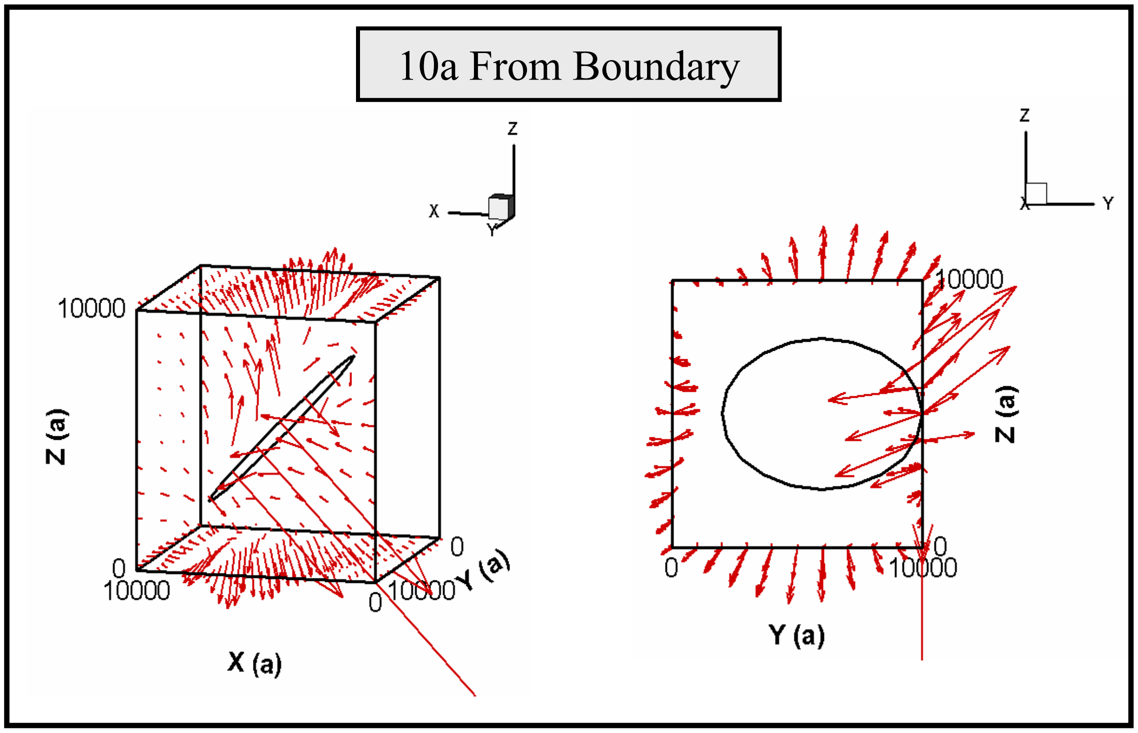


Figure 5.4.3.3: Image and resultant force distributions on shear loops 1000a, 250a, and 10a from the boundary. All vectors are to scale.

Figures 5.4.3.1 and 5.4.3.3 show the attractive nature of the image forces for distances greater than 10a and repulsive nature otherwise. Additionally, at 50 MPa the resultant forces are not great enough to overcome the Peierls force threshold, and thus the resultant forces are zero on all loop nodes. The image force distribution contributes heavily to the resultant force distribution at 10a from the boundary. It should also be noted that the image and resultant forces are all in plane. Figure 5.4.3.1A and 5.4.3.2A show that there is little if any out-of-plane force component in these force distributions.

To understand why the image forces become repulsive at close proximity to the boundary we must look at the surface tractions developed by the elastic stress field. Tractions were compared for this same configuration at 250a and 10a from the boundary. These tractions are depicted in figure 5.4.3.4. They show that there are large shear forces acting on the surface of the crystal and that these forces grow very large as the loop comes close to the boundary.

The deformation of the loop under the influence of the calculated resultant force distribution was computed. These results are presented in diagram 5.4.3.5 below. The loop deforms to establish an equilibrium position approximately 200 lattice constants from the y=10000a boundary.



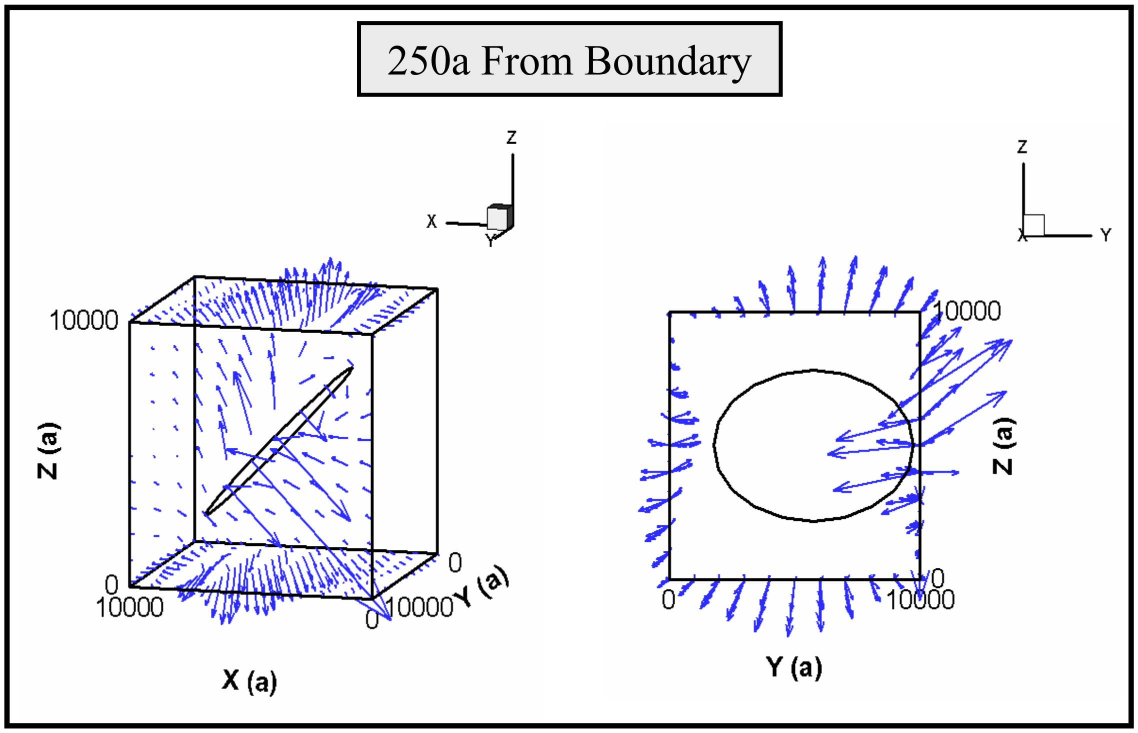


Figure 5.4.3.4: Surface tractions developed by the elastic field of a 4000a radius shear loop 10a and 250a from the boundary. The tractions developed nearest to the loop are shear forces. The largest traction magnitudes are as follows: 742.0x10^6 N/m (10a) and 27.0x10^6 N/m (250a). Tractions on certain planes have been removed for clarity.

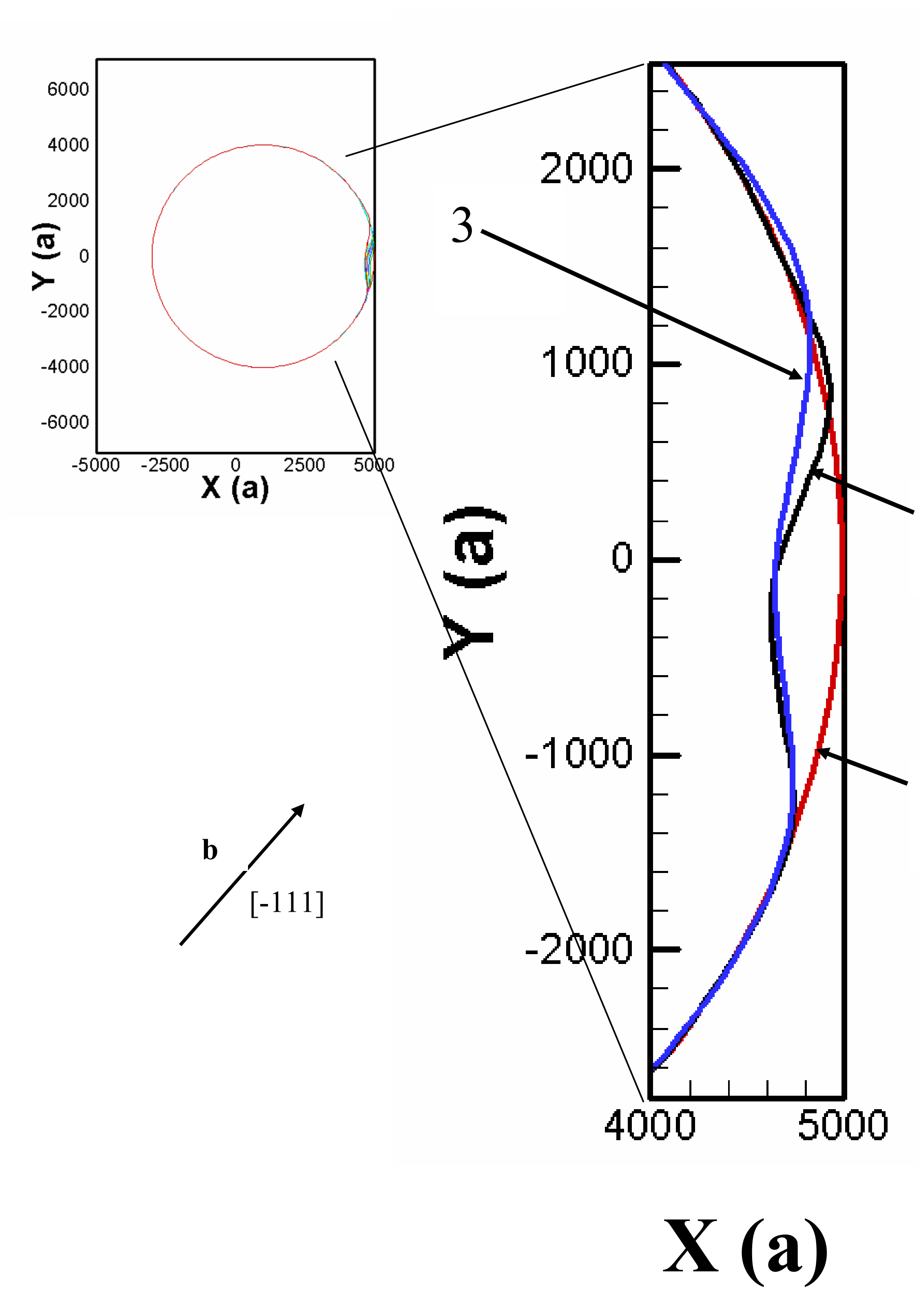


Figure 5.4.3.5: The initial loop perimeter (1), the updated perimeter after 20 time steps (2), and the updated loop perimeter after 116 time steps (3) (one time step =0.1 pico second). The Burgers vector is displayed.

5.5 Quantification of Image Force Effects

To better understand the quantitative effect that the proximity of the free surface to the loop has on the image force distribution, the following analyses were undertaken. For three radii (500a, 1000a, and 2000a), image and self-force distributions were computed for loops placed in varying proximity to the y=10000a plane, and the z=10000a plane. Both analyses involve the (101) [-111] slip system. For the z- proximity analysis the local origin of the slip plane is as follows: (7500, 5000, 7500). The material under consideration is a single crystal of BCC Fe with dimensions 10000a x 10000a x 10000a. Representative image force distributions for both analyses are depicted below.

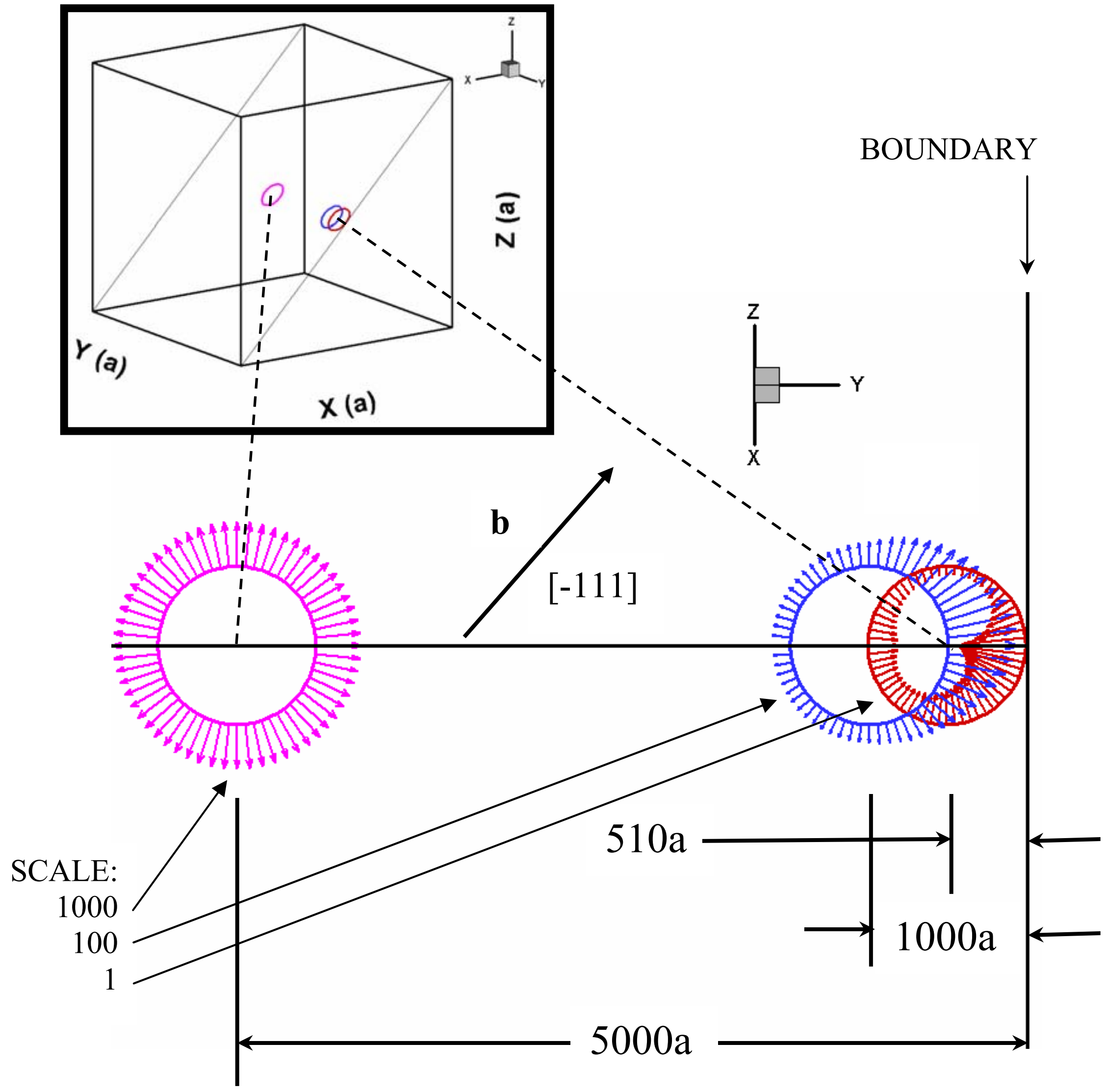


Figure 5.5.1: 3-d and 2-d representations of the image force distributions and loop geometry used in the y proximity analysis. Scale factors are as listed.

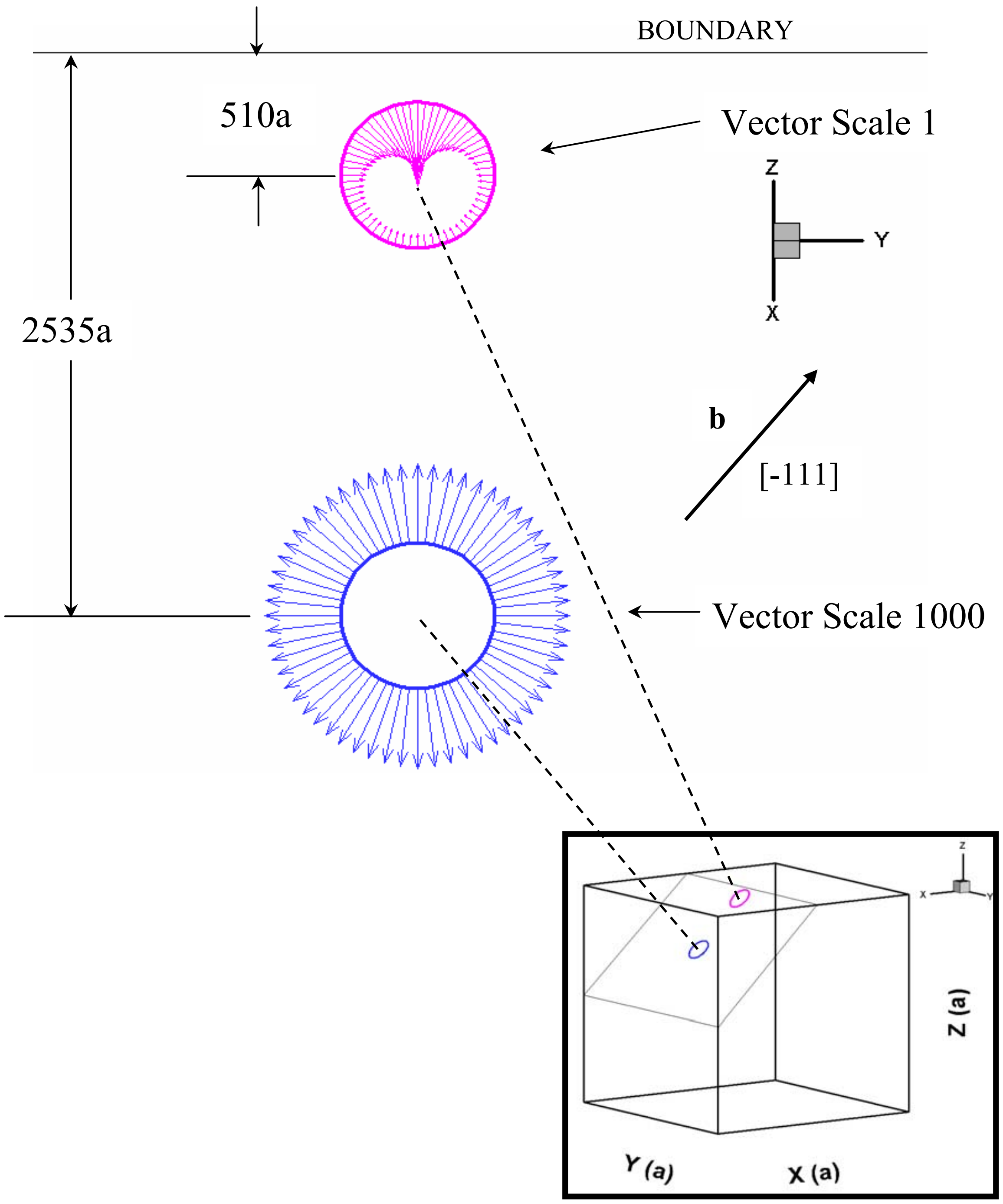


Figure 5.5.2: 2-d and 3-d depictions of the image force distribution on 500a radius loops placed at different locations with respect to the z = 10000a surface. The scaling factors are as indicated.

Self-force distributions are in-plane for both analyses, however the image forces have a large out-of-plane component for loops near the z = 10000a surface. To quantify the magnitude of the image force with respect to the proximity to the surface, two values had to be determined. For in-plane analysis of image force values, the largest nodal image force vector was projected onto the slip plane and its magnitude was determined. This value was then divided by the largest self force magnitude. The sign of this ratio was taken as negative for image force values directed toward the center of curvature of the loop, and positive for image force values directed away from the center of curvature for the loop. In this way repulsive image force distributions were defined as negative and attractive image force distributions as positive. For out-of-plane analysis of image force values associated with only the z-proximity analyses, the largest nodal image force vector component normal to the slip plane was determined. The magnitude of this value was divided by the largest self force magnitude. In this way, a normalized image force value was calculated to allow for a relationship between image force values and distance from the surface to be determined. The relationship between these quantities and the distance to the boundary are depicted in the following graphs for both the y-proximity analyses and the z-proximity analyses.

Figures 5.5.3, 5.5.4, 5.5.5 and 5.5.6 illustrate the dual attractive and repulsive nature of the image forces depending on proximity to either the y = 10000a surface, or the z = 10000a surface. For the y-proximity analyses, as the loops move closer to the boundary, the image forces become increasingly attractive until a maximum value is reached after which these attractive forces diminish, and become repulsive. This trend was observed for all three loop radii analyzed. Careful analysis of the results plotted in Figure 5.5.5 indicates the following. The distance from the surface at which the attractive nature of the image forces stops building and begins to diminish (inflection point) occurs at approximately 400a, 360a and 330a for the 1000a, 2000a, and 4000a diameter loop, respectively. The distance from the boundary at which the image forces on the loop transition from an attractive behavior to a repulsive one, occurs at approximately 200a, 140a, and 120a for the 1000a, 200a, and 4000a diameter loop, respectively.

The z-proximity analyses indicate that the attractive out-of-plane component to the image forces dominates over the repulsive in-plane component within close proximity of the boundary. As the loop is brought closer to the boundary, forces on the loop nodes closest to the boundary transition from in-plane attractive, to out-of-plane attractive. These image forces become perpendicular to the slip plane at a distance of approximately 500a from the surface. The image force distribution continues to rotate as the loop gets closer to the boundary (see figure 5.4.2.3).

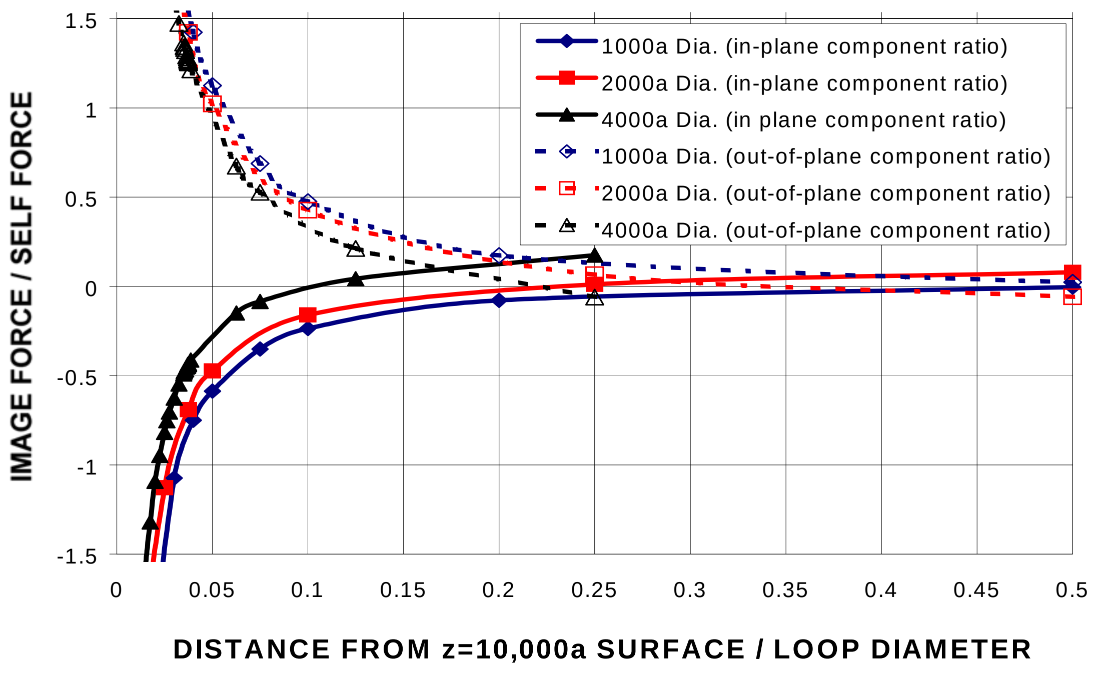


Figure 5.5.3: Magnitude of the largest nodal image force projected onto the slip plane, and in the direction normal to the slip plane, normalized by the magnitude of the largest nodal self force on a shear loops of various diameters. Slip system: (101) [-111].

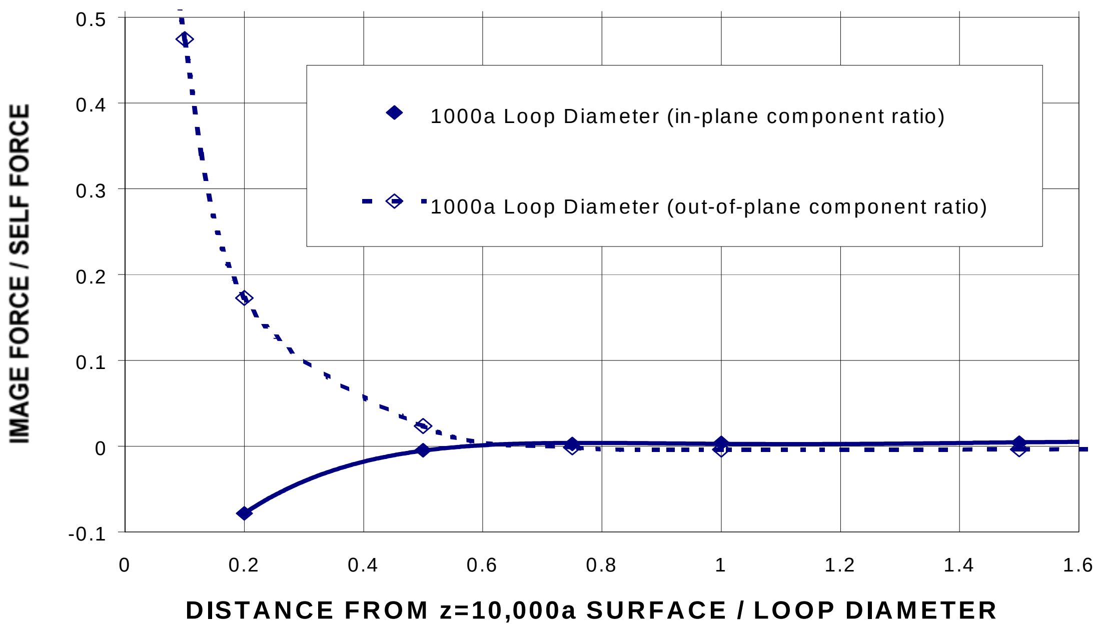


Figure 5.5.4: Magnitude of the largest nodal image force projected onto the slip plane, and in the direction normal to the slip plane, normalized by the magnitude of the largest nodal self force on a shear loop. Slip system: (101) [-111].

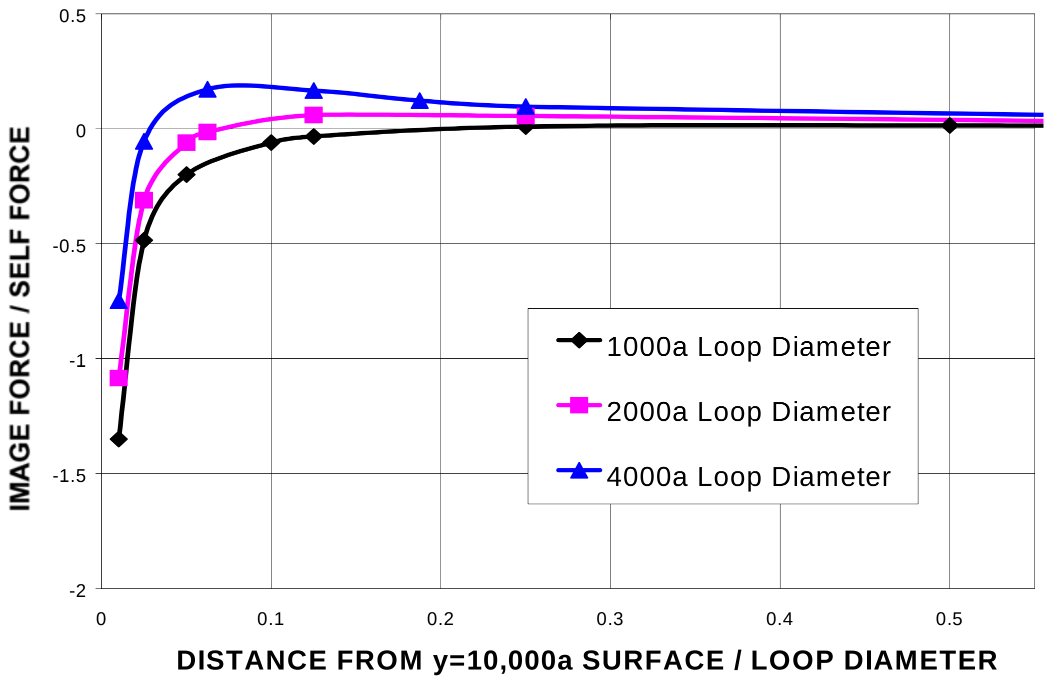


Figure 5.5.5: Magnitude of the largest nodal image force normalized by the magnitude of the largest nodal self force on shear loops of various diameters.

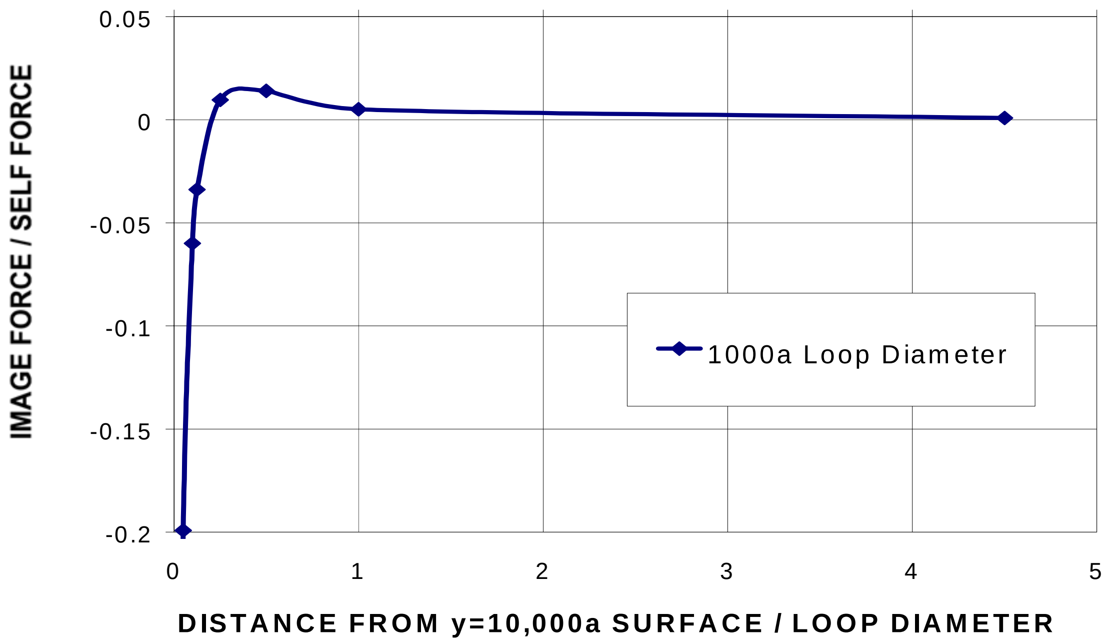


Figure 5.5.6: Magnitude of the largest nodal image force normalized by the magnitude of the largest nodal self force on a 1000a diameter shear loop.

5.6 Frank Reed Source Deformation Analyses

The deformed geometry of a Frank Reed source in BCC Fe under the influence of an applied stress was computed using the deformation program. In the first analysis depicted in figure 5.6.1 below, a stress of 125 MPa was applied. The FR source is located in the slip plane with origin (7500,5000,7500). The local coordinates of the source points are (-2000, 3000) and (2000,3000). The deformation program was run until an equilibrium geometry was achieved. This equilibrium geometry is consistent with the force distribution on a 2000a radius loop adjacent to the z = 10000a boundary. The shape is similar to that observed for the equilibrium geometry following deformation described by the z-proximity analyses. The equilibrium position of the source is approximately 163a from the surface of the crystal model.

In a second analysis, the ratio of edge to screw Peierls forces was manipulated to simulate and compare FR source deformation in "softer" crystal structures like the face centered cubic structure, with "harder" crystal structures like the body centered cubic crystal structure. To this end, two deformation results were compared. The first result depicted in figure 5.6.2C, shows the deformation in a material with a screw to edge Peierls force ratio of 2 to 1, a Peierls threshold of 1 x 10^{-6} μ (where μ is the shear modulus), and an applied stress of 200 MPa. This represents a material with a "low" screw to edge Peierls force ratio. The result depicted in figure 5.6.2D shows the deformation in a material with a screw to edge Peierls force ratio of 10 to 1, a Peierls threshold of 1 x 10^{-3} μ, and an applied stress of 700 MPa. . This represents a material with a “high” screw to edge Peierls force ratio. These results indicate that the deformation for the high ratio case evolves with significantly less curvature than the low ratio case.

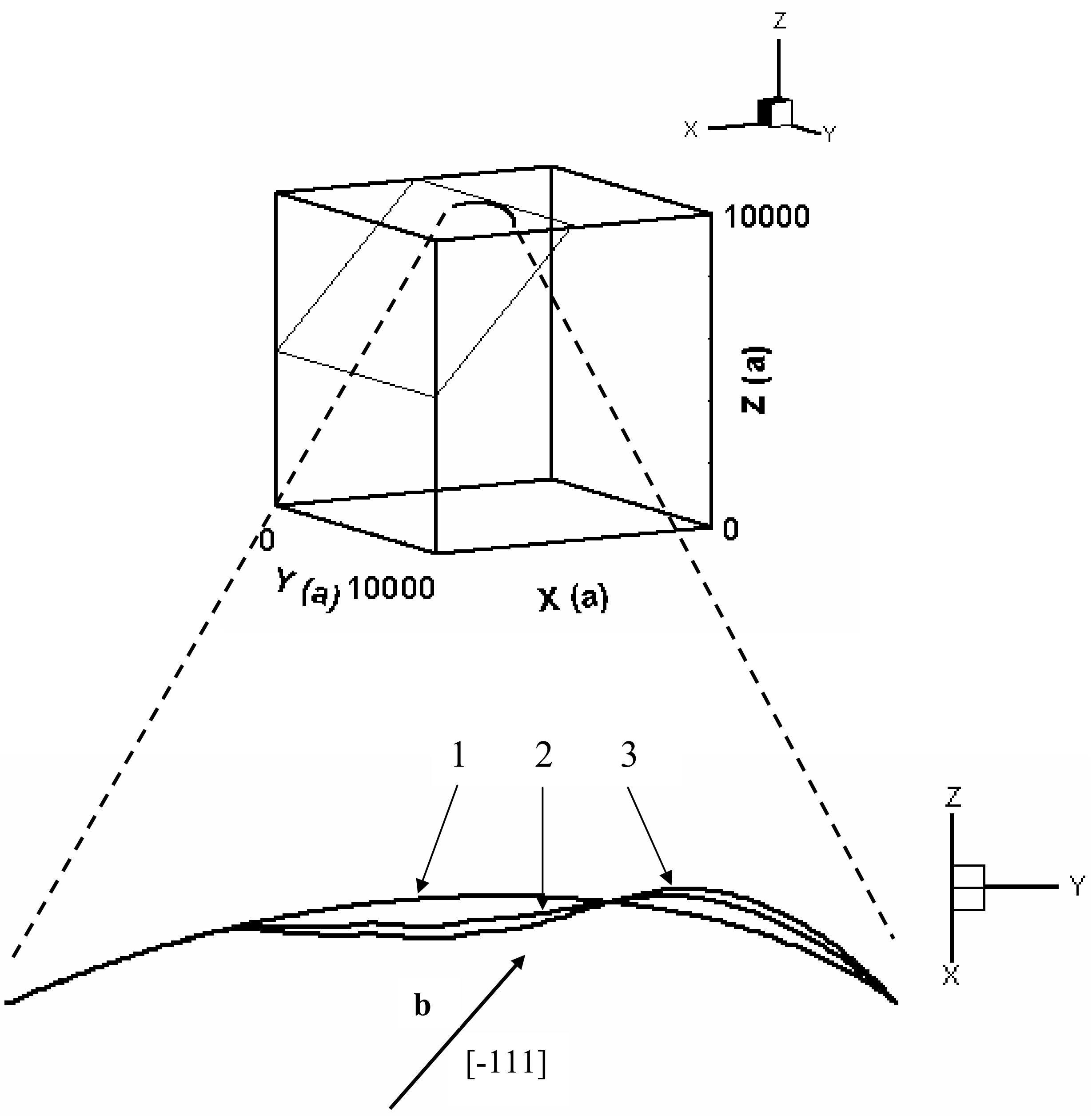
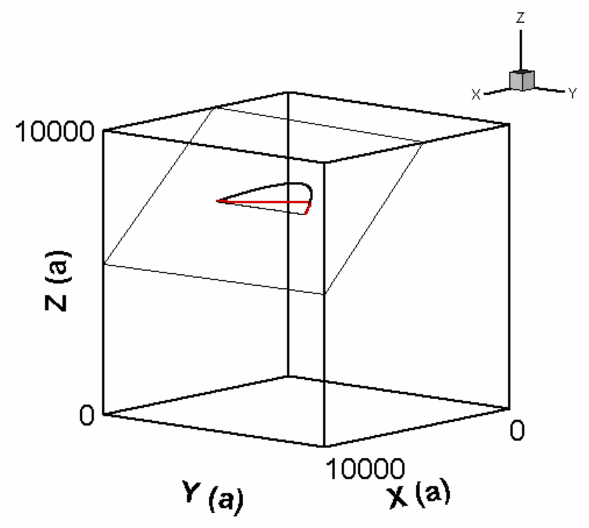
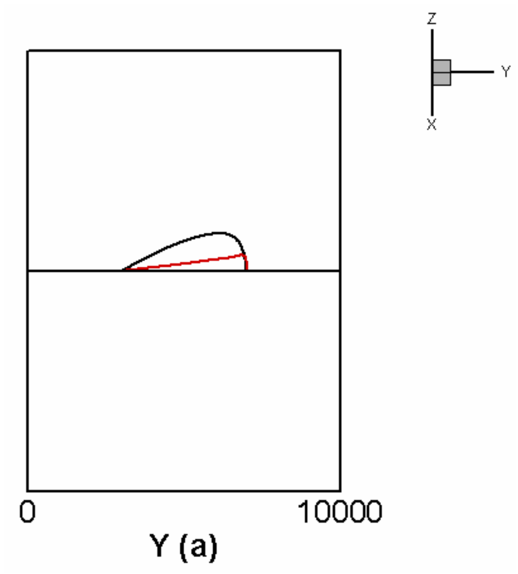


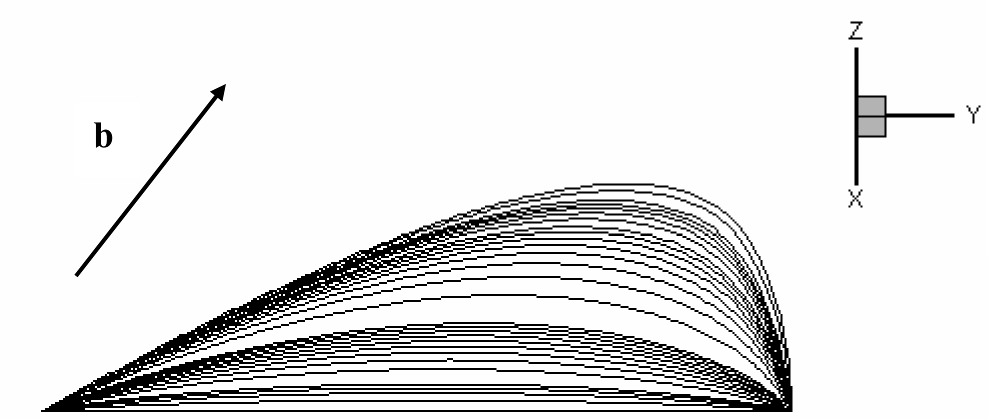
Figure 5.6.1: Isometric view of the model and FR source, and a 2-d view of the deformed geometry as viewed normal to the (101) [-111] slip system. Initial geometry (1), (3) is the equilibrium geometry under an applied stress σ_{xx} = 125 MPa.



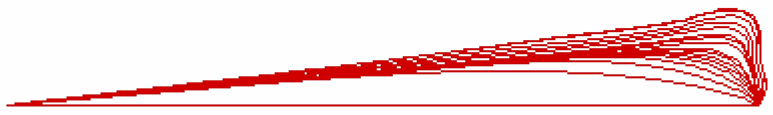
A



B



C



D

Figure 5.6.2: Isometric (A), and 2-d (B, C, and D) views of a FR source deforming. View C depicts deformation plotted for the "low" ratio case. View D depicts deformation plotted for the "high" ratio case. 4000 time steps.

5.7 Comments on Image Forces

Friedel makes several relevant comments with respect to image forces and the attractive/repulsive nature of these forces [39]. A dislocation is attracted toward its image, (toward a free surface) by a force computed as follows:

$$F = -\frac{\mu\, b^2}{4\,\pi\, L} \tag{5.7.1}$$

where L is the distance from the free surface, b is the Burgers vector and μ is the shear modulus. Furthermore, Friedel states that a dislocation should arrive perpendicular to the free surface in order to achieve the most stable configuration.

However, a dislocation in a medium that is separated from a free surface by a thin film as shown below, behaves quite differently.

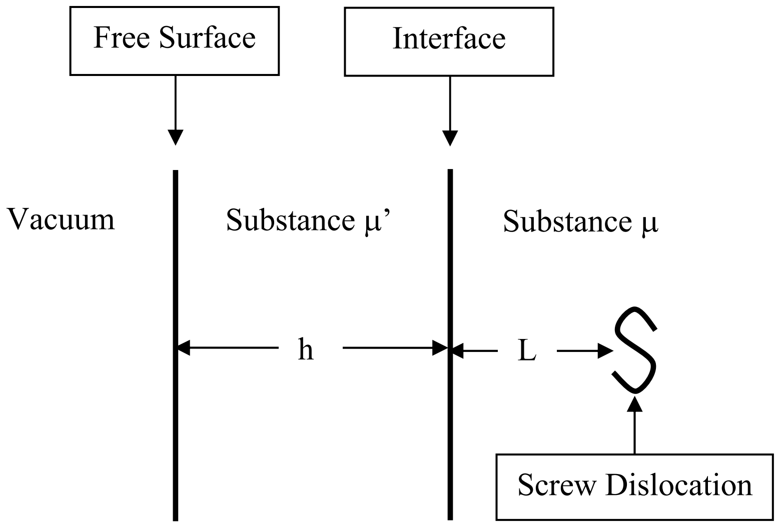
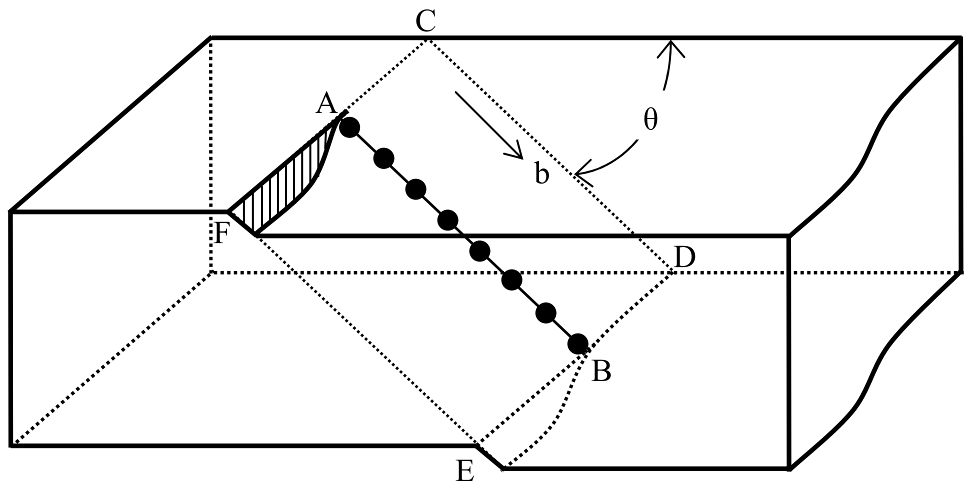


Figure 5.7.1: Free surface/interface model, (after [39]).

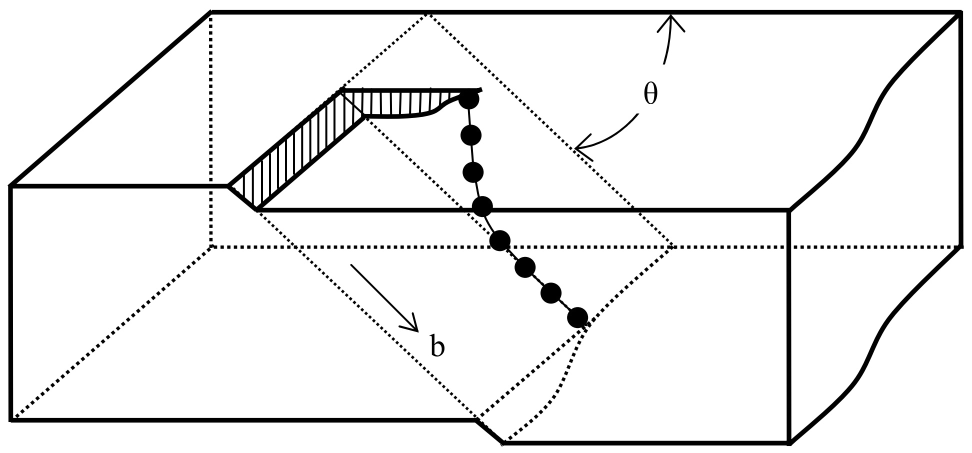
A screw dislocation will be drawn toward the free surface if $\mu' < \mu$. It will be repelled however, if $\mu' > \mu$ and $L << h$. For $\mu' > \mu$, and $L >> h$, the dislocation will be attracted toward the surface. This implies that there is an equilibrium position from the surface on the order of thickness h from the interface.

Nabarro echoes several of the comments made by Friedel [41]. The surface may act as a barrier to the escape of dislocations depending on the presence of surface films. However, once the film (i.e. an oxide layer) is removed, the dislocations migrate out of the surface. In ionic crystals such as potassium chloride, evidence supports the conclusion that image forces repel dislocations from breaking the surface. Although the single crystal model explored by this thesis does not involve a thin film layer, these comments seem appropriate to present based on the y-proximity analyses results. High shear stresses are present on the surface of the model (see figure 5.4.3.4). Similarly, high shear stresses are associated with some thin film/substrate interfaces due to factors like lattice constant mismatch, and coefficient of thermal expansion mismatch [53]. For this reason, the repulsive effect uncovered for normal slip plane/free surface orientations may be better understood through further exploration of the fundamental similarities in the stress distributions between the repulsive scenario described herein, and the thin film model proposed above.

Nabarro also states that the image forces may induce cross slip. Dislocations tend to meet the free surface normally because this minimizes their free energy [41]. Gilman and coworkers conducted experimental studies involving the measurement of dislocation velocities in LiF single crystals [49]. During the course of these studies, important observations of dislocation behavior were made. In the paper "The Mechanism of Surface Effects in Crystal Plasticity" (1961) [48], they propose a mechanism of dislocation motion under the influence of surface effects in a single crystal. The schematic of this mechanism is illustrated in figure 5.7.2 below.



(A)



(B)

Figure 5.7.2: Cross-slip of a screw dislocation adjacent to a surface in a crystal. Initial configuration (A), and the deformation of the dislocation line following cross-slip (B); (after [48]).

In figure 5.7.2A a screw dislocation (AB) lies in the glide plane (CDEF), which is at an angle θ with respect to the free surface. Although the dislocation typically would move in the direction of (AC), deformation through cross-slip (figure 5.7.2B) has been observed [49]. This mechanism of cross-slip was assumed by Gilman to be the result of energy minimization through the reduction in the length of the dislocation along the slip plane [49]. With the image force distribution observed in figure 5.4.2.3 above, it is obvious that this effect, which was postulated over 40 years ago, is the result of the image force distribution simulated in the z-proximity analyses.

The arguments stated above support several aspects of the results obtained in this study. As the dislocation loop comes within a close proximity of the surface in the z-proximity analysis, the image forces dominate the overall force distribution. The image force distribution is characterized by large out-of-plane force components that seek to pull the loop to the surface through the action of cross-slip. Furthermore, image forces repel dislocation loops approaching a free surface on a slip plane which is normal to the free surface. These results correlate well with arguments made by Nabarro, Gilman and Friedel.

6 SUMMARY AND CONCLUSIONS

The modeling and numerical results presented in this thesis seek to illustrate the nature of free surface effects on dislocations in metallic single crystals. Much of the work in Dislocation Dynamics that is presented in the literature testifies to the importance of these surface effects. However, little if any modeling has been completed which describes the precise nature of the force distributions induced by traction free boundary conditions.

In this work, force distributions on a shear dislocation loop in the (101) [-111] slip system in an Fe single crystal were computed and plotted. The deformation resulting from these forces was also computed and plotted. Furthermore, the effects of image forces and the variation of the screw to edge Peierls force ratio on the deformation of a FR source was computed and plotted. Based on these results the following conclusions can be drawn.

1. A systematic method for coupling Dislocation Dynamics numerical modeling with FEM has been successfully developed for 3-D crystal models using the Superposition approach. Traction free boundary conditions are achieved through the use of this approach.

2. Image forces act to either attract or repel the loop from the boundary depending on the distance of the loop from the boundary, and the orientation of the slip plane with

respect to the nearest free surface. For slip planes at an oblique angle to a free surface, the image force distribution on the closest portions of the loop to the surface are characterized by a large out-of-plane forces. These forces are characterized by components perpendicular to the slip plane which act to pull the loop toward the surface through the mechanism of cross-slip. These forces are also characterized by an in-plane component which repels the closed loop from reaching the surface at the intersection of the slip plane with the free surface. For slip planes that are normal to the free surface, image force distributions on dislocation loops have negligible out-of-plane components. These image forces repel portions of the loop closest to the boundary, from the boundary.

3. The Gilman-Nabarro mechanism of cross-slip has been verified and extended to closed dislocation loops.

4. Image forces dominate the resultant force distribution for portions of the loop close to the free surface. The distance from the boundary at which the image forces become significant can be estimated based on the image force/self force ratio. This distance varies with respect to loop radius. Loops with large radii are effected by image forces deeper into the single crystal than are loops with small radii.

5. An equilibrium loop position close to the free surface can be achieved for dislocations on slip planes which are perpendicular to the surface. This equilibrium

effect was only observed for carefully determined applied stress values. For stress values too high, the loop expands to the surface. For stress values too low, forces on the loop act to cause it to contract, depending on the radius of the loop. For small loops, the self forces dominate and cause contraction. Thus, the expansion or contraction of the loop is dependent on a critical applied stress, radius of curvature, and proximity from the boundary.

6. Due to the coarseness of the FEM mesh, loops close to the edge of the model experience significant "edge effects". These effects are characterized by the overestimation of the out-of-plane component of the image forces. These effects can be moderated by increasing the concentration of the FEM mesh. The repulsive nature of the image forces for certain slip plane/free surface orientations is not however a numerical artifact relating to the mesh size.

7. Four aspects of the current computational methodology require improvement and or modification:
   i. The edge effects can be mitigated by increasing the concentration of the mesh at the surface of the FEM model, but leaving the mesh sparse in the bulk. This will allow for computational efficiency and increase accuracy.
   ii. Alteration of the FEM meshing will require the Seeker subroutine to be altered. This should not be a deterrent. The Seeker subroutine requires modifications to make it more efficient.

iii. The Deformation program should be improved to allow for faster computation, and also to allow for loop nodes to freeze when they appear at the surface of the single crystal model. This will greatly aid in future modeling efforts.

iv. Deformation due to the mechanism of cross-slip should be incorporated into the numerical model.

APPENDIX A: The Microplasticity Fortran90 Code

```
MODULE vectors

   TYPE vector
     DOUBLE PRECISION,DIMENSION(3)::v
   END TYPE vector

   TYPE matrix
     TYPE(vector),DIMENSION(3)::v
   END TYPE matrix

   TYPE tensor
     TYPE(matrix),DIMENSION(3)::v
   END TYPE tensor

   INTERFACE OPERATOR(+)
     MODULE PROCEDURE v1_plus_v2;MODULE PROCEDURE m1_plus_m2
     MODULE PROCEDURE t1_plus_t2
   END INTERFACE
   INTERFACE OPERATOR(-)
     MODULE PROCEDURE v1_minus_v2;MODULE PROCEDURE m1_minus_m2
     MODULE PROCEDURE t1_minus_t2
   END INTERFACE
   INTERFACE OPERATOR(*)
     MODULE PROCEDURE v1_dot_v2;MODULE PROCEDURE real_times_v1
     MODULE PROCEDURE m1_mul_v2;MODULE PROCEDURE m1_mul_m2
     MODULE PROCEDURE t1_mul_v2;MODULE PROCEDURE real_times_t1
     MODULE PROCEDURE real_times_m1
   END INTERFACE
   INTERFACE OPERATOR(/)
     MODULE PROCEDURE v1_div_real;MODULE PROCEDURE m1_div_real
     MODULE PROCEDURE t1_div_real
     MODULE PROCEDURE v1_outerproduct_v2
     MODULE PROCEDURE m1_outerproduct_v2
     MODULE PROCEDURE v1_outerproduct_m2
   END INTERFACE
   INTERFACE OPERATOR(//)
     MODULE PROCEDURE m1_outerproduct1_v2
   END INTERFACE
   INTERFACE OPERATOR(**)
     MODULE PROCEDURE v1_cross_v2
     MODULE PROCEDURE theta_rotate_v1
   END INTERFACE
   CONTAINS

   TYPE(vector) FUNCTION v1_minus_v2(a,b)
   TYPE(vector), INTENT(IN)::a,b
   DO i=1,3;v1_minus_v2%v(i)=a%v(i)-b%v(i);END DO
   END FUNCTION v1_minus_v2

   TYPE(matrix) FUNCTION m1_minus_m2(a,b)
   TYPE(matrix), INTENT(IN)::a,b
   DO i=1,3;DO j=1,3
     m1_minus_m2%v(i)%v(j)=a%v(i)%v(j)-b%v(i)%v(j)
   END DO;END DO
   END FUNCTION m1_minus_m2

   TYPE(tensor) FUNCTION t1_minus_t2(a,b)
   TYPE(tensor), INTENT(IN)::a,b
```

```
DO i=1,3;DO j=1,3;DO k=1,3
  t1_minus_t2%v(i)%v(j)%v(k)=a%v(i)%v(j)%v(k)-b%v(i)%v(j)%v(k)
END DO;END DO;END DO
END FUNCTION t1_minus_t2

TYPE(vector) FUNCTION v1_plus_v2(a,b)
TYPE(vector), INTENT(IN)::a,b
DO i=1,3;v1_plus_v2%v(i)=a%v(i)+b%v(i);END DO
END FUNCTION v1_plus_v2

TYPE(matrix) FUNCTION m1_plus_m2(a,b)
TYPE(matrix), INTENT(IN)::a,b
DO i=1,3;DO j=1,3
  m1_plus_m2%v(i)%v(j)=a%v(i)%v(j)+b%v(i)%v(j)
END DO;END DO
END FUNCTION m1_plus_m2

TYPE(tensor) FUNCTION t1_plus_t2(a,b)
TYPE(tensor), INTENT(IN)::a,b
DO i=1,3;DO j=1,3;DO k=1,3
  t1_plus_t2%v(i)%v(j)%v(k)=a%v(i)%v(j)%v(k)+b%v(i)%v(j)%v(k)
END DO;END DO;END DO
END FUNCTION t1_plus_t2

TYPE(vector) FUNCTION real_times_v1(a,b)
DOUBLE PRECISION,INTENT(IN)::a
TYPE(vector), INTENT(IN)::b
DO i=1,3;real_times_v1%v(i)=a*b%v(i);END DO
END FUNCTION real_times_v1

TYPE(matrix) FUNCTION real_times_m1(a,b)
DOUBLE PRECISION,INTENT(IN)::a
TYPE(matrix), INTENT(IN)::b
DO i=1,3;DO j=1,3
  real_times_m1%v(i)%v(j)=a*b%v(i)%v(j)
END DO;END DO
END FUNCTION real_times_m1

TYPE(tensor) FUNCTION real_times_t1(a,b)
DOUBLE PRECISION,INTENT(IN)::a
TYPE(tensor), INTENT(IN)::b
DO i=1,3;DO j=1,3;DO k=1,3
  real_times_t1%v(i)%v(j)%v(k)=a*b%v(i)%v(j)%v(k)
END DO;END DO;END DO
END FUNCTION real_times_t1

TYPE(vector) FUNCTION v1_div_real(a,b)
DOUBLE PRECISION,INTENT(IN)::b
TYPE(vector), INTENT(IN)::a
DO i=1,3;v1_div_real%v(i)=a%v(i)/b;END DO
END FUNCTION v1_div_real

TYPE(matrix) FUNCTION m1_div_real(a,b)
DOUBLE PRECISION,INTENT(IN)::b
TYPE(matrix), INTENT(IN)::a
DO i=1,3;DO j=1,3
  m1_div_real%v(i)%v(j)=a%v(i)%v(j)/b
END DO;END DO
END FUNCTION m1_div_real

TYPE(tensor) FUNCTION t1_div_real(a,b)
DOUBLE PRECISION,INTENT(IN)::b
TYPE(tensor), INTENT(IN)::a
DO i=1,3;DO j=1,3;DO k=1,3
  t1_div_real%v(i)%v(j)%v(k)=a%v(i)%v(j)%v(k)/b
END DO;END DO;END DO
END FUNCTION t1_div_real

TYPE(matrix) FUNCTION v1_outerproduct_v2(a,b)
TYPE(vector), INTENT(IN)::a,b
DO i=1,3;DO j=1,3
  v1_outerproduct_v2%v(i)%v(j)=a%v(i)*b%v(j)
END DO;END DO
END FUNCTION v1_outerproduct_v2

TYPE(tensor) FUNCTION m1_outerproduct_v2(a,b)
TYPE(matrix), INTENT(IN)::a
TYPE(vector), INTENT(IN)::b
DO i=1,3;DO j=1,3;DO k=1,3
  m1_outerproduct_v2%v(i)%v(j)%v(k)=a%v(i)%v(j)*b%v(k);
END DO;END DO;END DO
END FUNCTION m1_outerproduct_v2

TYPE(tensor) FUNCTION m1_outerproduct1_v2(a,b)
TYPE(matrix), INTENT(IN)::a
TYPE(vector), INTENT(IN)::b
DO i=1,3;DO j=1,3;DO k=1,3
  m1_outerproduct1_v2%v(i)%v(j)%v(k)=a%v(k)%v(i)*b%v(j);
END DO;END DO;END DO
END FUNCTION m1_outerproduct1_v2

TYPE(tensor) FUNCTION v1_outerproduct_m2(a,b)
TYPE(vector), INTENT(IN)::a
TYPE(matrix), INTENT(IN)::b
DO i=1,3;DO j=1,3;DO k=1,3
  v1_outerproduct_m2%v(i)%v(j)%v(k)=b%v(j)%v(k)*a%v(i);
END DO;END DO;END DO
END FUNCTION v1_outerproduct_m2

TYPE(vector) FUNCTION VECT(a,b)
TYPE(tensor), INTENT(IN)::a
INTEGER::b
DO i=1,3
  SELECT CASE(b)
  CASE(1)
    VECT%v(i)=a%v(i)%v(1)%v(1)+a%v(i)%v(2)%v(2)+a%v(i)%v(3)%v(3)
  CASE(2)
    VECT%v(i)=a%v(1)%v(i)%v(1)+a%v(2)%v(i)%v(2)+a%v(3)%v(i)%v(3)
  CASE(3)
    VECT%v(i)=a%v(1)%v(1)%v(i)+a%v(2)%v(2)%v(i)+a%v(3)%v(3)%v(i)
  END SELECT
END DO
END FUNCTION VECT

DOUBLE PRECISION FUNCTION v1_dot_v2(a,b)
TYPE(vector), INTENT(IN)::a,b
v1_dot_v2=a%v(1)*b%v(1)+a%v(2)*b%v(2)+a%v(3)*b%v(3)
END FUNCTION v1_dot_v2

TYPE(vector) FUNCTION v1_cross_v2(a,b)
TYPE(vector), INTENT(IN)::a,b
v1_cross_v2%v(1)=a%v(2)*b%v(3)-b%v(2)*a%v(3)
v1_cross_v2%v(2)=a%v(3)*b%v(1)-b%v(3)*a%v(1)
v1_cross_v2%v(3)=a%v(1)*b%v(2)-b%v(1)*a%v(2)
END FUNCTION v1_cross_v2

TYPE(vector) FUNCTION theta_rotate_v1(a,b)
DOUBLE PRECISION, INTENT(IN)::a
TYPE(vector), INTENT(IN)::b
TYPE(vector)::c
c%v(1)=0;c%v(2)=0;c%v(3)=1
theta_rotate_v1=dcos(a)*b+dsin(a)*c**b
```

```
END FUNCTION theta_rotate_v1

TYPE(vector) FUNCTION m1_mul_v2(a,b)
TYPE(matrix), INTENT(IN)::a
TYPE(vector), INTENT(IN)::b
DO i=1,3;m1_mul_v2%v(i)=a%v(i)*b;END DO
END FUNCTION m1_mul_v2

TYPE(matrix) FUNCTION TRANS(a)
TYPE(matrix),INTENT(IN)::a
DO i=1,3;DO j=1,3;TRANS%v(i)%v(j)=a%v(j)%v(i);END DO;END DO
END FUNCTION TRANS

TYPE(matrix) FUNCTION m1_mul_m2(a,b)
TYPE(matrix), INTENT(IN)::a,b
DO i=1,3;m1_mul_m2%v(i)=TRANS(b)*a%v(i);END DO
END FUNCTION m1_mul_m2

TYPE(matrix) FUNCTION t1_mul_v2(a,b)
TYPE(tensor), INTENT(IN)::a
TYPE(vector), INTENT(IN)::b
DO i=1,3;t1_mul_v2%v(i)=a%v(i)*b;END DO
END FUNCTION t1_mul_v2

DOUBLE PRECISION FUNCTION MAG(A)
TYPE(vector),INTENT(IN)::A
MAG=DSQRT(A%v(1)**2+A%v(2)**2+A%v(3)**2)
END FUNCTION MAG

TYPE(vector) FUNCTION UV(A)
TYPE(vector),INTENT(IN)::A
UV=A/MAG(A)
END FUNCTION UV

END MODULE vectors

!===============================================================
MODULE VARIABLES
USE VECTORS
IMPLICIT NONE

!MAX_LOOP: MAXIMUM NUMBER OF LOOPS
!MAX_NODE: MAXIMUM NUMBER OF NODES FOR A LOOP
!NPLOT: NUMBER OF PLOTTING POINTS FOR A SEGMENT
!OMAX: MAXIMUM NUMBER OF OBSTACLES
!N_NODE: Number of nodes for loops without obstacles

CHARACTER(LEN=40)::INTERACTION_TYPE,CALCULATION_TYPE,SEG_TYPE,
+                 CAD_PLOT,ELASTIC_FIELD_TYPE,FORCE_TYPE,
+                 FORCE_KIND,DEFECT_TYPE
!   SEG_TYPE=LINEAR;ARC; CUBIC, QUINTIC , COMP_ARC

DOUBLE PRECISION::MIN_NR,MD,DIS1,CU2,R0,ROBS,RMAX1,ALPHA,CRIT,
+                SIGMA,RC,NP,MS,DIS, APPLIED_SIG,
+                MU,NU,LATTICE,TEMP, A_CUBE, MOBILITY,
+                PI,CUR1,CU1,DENSITY,NR1,H,G,AS, RMIN, U,
+                NODE_AREA,INTERVAL,INTERVAL2, FACT2, FACT3,
+                DEFECT_RADIUS,NEAREST_DIST,ATOLL,RTOL,DTIME

INTEGER::I_TIME,N_TIMES,NLOOP_TEM,MN,OMAX,NPLOT, IC, N_Q_POINTS,
+ VALSPLIT,ILSPLIT,ISPLIT,VALANN,I_ANN,J_ANN,IL_ANN,JL_ANN,I_TIME1,
+ III,I_P, MAX_QUAD, MAX_LOOP, MAX_PLANE, MAX_NODE, IL, N_LOOP_TOT,
+ ID_JOG, ID_NODE_L, ID_NODE_G, N_QUAD, N_PLANE, ID_SEG,N_LOOP_OBS,
+ N_DIPOLE

INTEGER,DIMENSION(2)::NQ
```

```
 TYPE(VECTOR)::R_PLOT,R1,RT,V,GV,VEC, DISPLACE, ARB, QG,LOOP_NORMAL

 TYPE (VECTOR)::FEM_NODE

 TYPE(VECTOR),DIMENSION(2,180)::QD,QT

 TYPE(MATRIX)::D,ZERO,SIG_APP, SIG, STRAIN, SIG_L

 TYPE (MATRIX), DIMENSION(8):: CAPSULE_STRESS

 TYPE (VECTOR), DIMENSION(8)::CAPSULE

 !MAX_PLANE
 INTEGER,ALLOCATABLE,DIMENSION(:)::NLOOP,NLOOP1,NOBS,N_L,JOIN,
+                               VALSTOP1
 DOUBLE PRECISION,ALLOCATABLE,DIMENSION(:)::NR
 TYPE(VECTOR),ALLOCATABLE,DIMENSION(:)::ORIGIN,MILLER
 TYPE(MATRIX),ALLOCATABLE,DIMENSION(:)::ES

 INTEGER,DIMENSION(4)::IM,KN
 DOUBLE PRECISION,DIMENSION(4)::M,UM


 !MAX_PLANE,MAX_LOOP
 INTEGER,ALLOCATABLE,DIMENSION(:,:)::NSEG,LOOPTYPE1,VALSTOP,
+     LOOPCHANGE,LOOPTYPE,VALID, N_NODE, NTEM, N_Q
 TYPE(VECTOR),ALLOCATABLE,DIMENSION(:,:)::BURGERS,BURGERS_OB
 DOUBLE PRECISION, ALLOCATABLE, DIMENSION(:,:)::RR0, PERIMETER

 !MAX_PLANE,OMAX
 TYPE(VECTOR),ALLOCATABLE,DIMENSION(:,:)::OBS
 INTEGER,ALLOCATABLE,DIMENSION(:,:)::VALOBS

 !MAX_PLANE,MAX_LOOP,0:MAX_NODE
 TYPE(VECTOR),ALLOCATABLE,DIMENSION(:,:,:)::PL,TL,NL,PTEM,TTEM,NT
 INTEGER,ALLOCATABLE,DIMENSION(:,:,:)::CUSP,CUSPTEM
 DOUBLE PRECISION,ALLOCATABLE,DIMENSION(:,:,:)::T1,T2,N1,N2,CUR,
+    CURTEM,CU

 !MAX_PLANE,MAX_LOOP,3
 INTEGER,ALLOCATABLE,DIMENSION(:,:,:)::PROP_DIPOLE
 !1:NUMBER OF DIPOLE 2:I_P 3:IL

 !MAX_PLANE,MAX_LOOP,180
 TYPE(VECTOR),ALLOCATABLE,DIMENSION(:,:,:)::Q_E,Q_D,Q_T


 !MAX_PLANE,MAX_LOOP,MAX_LOOP
 DOUBLE PRECISION,ALLOCATABLE,DIMENSION(:,:,:)::PMD

 !MAX_PLANE,MAX_LOOP,MAX_LOOP
 INTEGER,ALLOCATABLE,DIMENSION(:,:,:)::L_L

 !MAX_PLANE,MAX_LOOP,MAX_NODE
 TYPE (VECTOR),ALLOCATABLE,DIMENSION(:,:,:)::  PG, TG,
+  NG, FP, FT, FN ,PS,PE,CC,CC1
 DOUBLE PRECISION,ALLOCATABLE,DIMENSION(:,:,:)::SHEAR,RR1
 INTEGER,ALLOCATABLE,DIMENSION(:,:,:)::VALID1,NL1

 !MAX_PLANE,MAX_PLANE,MAX_LOOP,MAX_LOOP
 DOUBLE PRECISION,ALLOCATABLE,DIMENSION(:,:,:,:)::ENERGY

 INTEGER,ALLOCATABLE,DIMENSION(:,:)::ELEMENT

 TYPE(VECTOR)::B_Q,MILLER_Q
 TYPE(MATRIX)::SIG_Q,SIG_ESTIMATED
 END MODULE VARIABLES
 !**************************************************************!
 !**************************************************************!
 PROGRAM MICROPLASTICITY

 USE VARIABLES
 USE VECTORS

 IMPLICIT NONE

 CALL OPEN_IO_FILES

 CALL DEFAULTS

 !READ Input Geometry files

 CALL DATA_READER

 !Set initial values

 CALL INITIAL

 !Start calculations

 SELECT CASE(CALCULATION_TYPE)
 CASE("INTERACTION")

   CALL COMPUTE_INTERACTION
   WRITE(*,*)"INTERACTION CALCULATION IS FINISHED."

 CASE("DEFORMATION")

   CALL COMPUTE_DEFORMATION
   WRITE(*,*)"DEFORMATION CALCULATION IS FINISHED."
 END SELECT

 END PROGRAM MICROPLASTICITY
 !**************************************************************!
 !**************************************************************!
 SUBROUTINE ADD_POINT(ILL,I,UU)
 USE VECTORS;USE VARIABLES
 INTEGER,INTENT(IN)::ILL,I;DOUBLE PRECISION,INTENT(IN)::UU
 NTEM(I_P,ILL)=NTEM(I_P,ILL)+1
 IF (NR(I_P)==1.0D0) THEN
   CUSPTEM(I_P,ILL,NTEM(I_P,ILL))=0
   IF(((CUSP(I_P,ILL,I)==-2).OR.(CUSP(I_P,ILL,I)==2)).AND.
+  (UU==0.0D0)) THEN
    CUSPTEM(I_P,ILL,NTEM(I_P,ILL))=CUSP(I_P,ILL,I)
   END IF
 END IF
 CALL PLOT(ILL,I,UU)
 PTEM(I_P,ILL,NTEM(I_P,ILL))=R_PLOT
 IF (LOOPTYPE(I_P,ILL)==1) THEN
   IF (((I/=1).OR.(UU/=0.0D0)).AND.(I/=N_NODE(I_P,ILL))) THEN
    CALL DISPLACEMENT(ILL,I,UU)
   END IF
 ELSE
   CALL DISPLACEMENT(ILL,I,UU)
 END IF
 END SUBROUTINE ADD_POINT

 !**************************************************************!
                 SUBROUTINE ALLOCATE_DIMENSIONS

 USE VARIABLES
```

```
 USE VECTORS

 ALLOCATE(NLOOP(MAX_PLANE),NLOOP1(MAX_PLANE),NOBS(MAX_PLANE),
+        ORIGIN(MAX_PLANE),MILLER(MAX_PLANE),ES(MAX_PLANE),
+        JOIN(MAX_PLANE),VALSTOP1(MAX_PLANE),NR(MAX_PLANE))


 ALLOCATE(NSEG(MAX_PLANE,MAX_LOOP),LOOPTYPE1(MAX_PLANE,MAX_LOOP),
+        VALSTOP(MAX_PLANE,MAX_LOOP),LOOPTYPE(MAX_PLANE,MAX_LOOP),
+        LOOPCHANGE(MAX_PLANE,MAX_LOOP),VALID(MAX_PLANE,MAX_LOOP),
+        N_NODE(MAX_PLANE,MAX_LOOP),NTEM(MAX_PLANE,MAX_LOOP),
+        BURGERS(MAX_PLANE,MAX_LOOP),N_Q(MAX_PLANE,MAX_LOOP),
+        RR0(MAX_PLANE,MAX_LOOP),PERIMETER(MAX_PLANE,MAX_LOOP))

 ALLOCATE(OBS(MAX_PLANE,OMAX),VALOBS(MAX_PLANE,OMAX))

 ALLOCATE(PL(MAX_PLANE,MAX_LOOP,0:MAX_NODE),
+        TL(MAX_PLANE,MAX_LOOP,0:MAX_NODE),
+        NL(MAX_PLANE,MAX_LOOP,0:MAX_NODE),
+        PTEM(MAX_PLANE,MAX_LOOP,0:MAX_NODE),
+        TTEM(MAX_PLANE,MAX_LOOP,0:MAX_NODE),
+        NT(MAX_PLANE,MAX_LOOP,0:MAX_NODE),
+        CUSP(MAX_PLANE,MAX_LOOP,0:MAX_NODE),
+        CUSPTEM(MAX_PLANE,MAX_LOOP,0:MAX_NODE),
+        T1(MAX_PLANE,MAX_LOOP,0:MAX_NODE),
+        T2(MAX_PLANE,MAX_LOOP,0:MAX_NODE),
+        N1(MAX_PLANE,MAX_LOOP,0:MAX_NODE),
+        N2(MAX_PLANE,MAX_LOOP,0:MAX_NODE),
+        CUR(MAX_PLANE,MAX_LOOP,0:MAX_NODE),
+        CURTEM(MAX_PLANE,MAX_LOOP,0:MAX_NODE),
+        CU(MAX_PLANE,MAX_LOOP,0:MAX_NODE))

 ALLOCATE(Q_E(MAX_PLANE,MAX_LOOP,180),Q_D(MAX_PLANE,MAX_LOOP,180),
+        Q_T(MAX_PLANE,MAX_LOOP,180),
+        PROP_DIPOLE(MAX_PLANE,MAX_LOOP,2))

 ALLOCATE(L_L(MAX_PLANE,MAX_LOOP,MAX_LOOP),
+        PMD(MAX_PLANE,MAX_LOOP,MAX_LOOP))

 ALLOCATE(PG(MAX_PLANE,MAX_LOOP,MAX_NODE),
+        TG(MAX_PLANE,MAX_LOOP,MAX_NODE),
+        NG(MAX_PLANE,MAX_LOOP,MAX_NODE),
+        FP(MAX_PLANE,MAX_LOOP,MAX_NODE),
+        FT(MAX_PLANE,MAX_LOOP,MAX_NODE),
+        FN(MAX_PLANE,MAX_LOOP,MAX_NODE),
+        PS(MAX_PLANE,MAX_LOOP,MAX_NODE),
+        PE(MAX_PLANE,MAX_LOOP,MAX_NODE),
+        CC(MAX_PLANE,MAX_LOOP,MAX_NODE),
+        CC1(MAX_PLANE,MAX_LOOP,MAX_NODE),
+        SHEAR(MAX_PLANE,MAX_LOOP,MAX_NODE),
+        RR1(MAX_PLANE,MAX_LOOP,MAX_NODE),
+        VALID1(MAX_PLANE,MAX_LOOP,MAX_NODE),
+        NL1(MAX_PLANE,MAX_LOOP,MAX_NODE))

 ALLOCATE(ENERGY(MAX_PLANE,MAX_PLANE,MAX_LOOP,MAX_LOOP))

 ALLOCATE(ELEMENT(MAX_NODE,2))
                    END SUBROUTINE ALLOCATE_DIMENSIONS

        !************************************************************!

            SUBROUTINE ANNIHILATE(IL1,IL2)
 USE VECTORS;USE VARIABLES
 INTEGER,INTENT(IN)::IL1,IL2
 INTEGER::Q1,Q2,I,I1,I2,I3,J,J1,J2,J3,J4,JOINCASE,L1,L2,Q3,Q4
 DOUBLE PRECISION::TEM,DOT1,DOT2,DOT3,DOT4,DIST1,DIST2,DIST3,DIST4,
+                  UU,ADJ,RTEM
 TYPE(VECTOR)::X,TA,TA1,PETEM
 JOIN(I_P)=1;CU2=0.4
 IF (IL1==IL2) THEN;JOINCASE=1
 ELSE;JOINCASE=2+LOOPTYPE(I_P,IL1)+LOOPTYPE(I_P,IL2);END IF
 SELECT CASE(JOINCASE)
 CASE(1) !OPEN-ITSELF
 NLOOP(I_P)=NLOOP(I_P)+1;LOOPTYPE(I_P,NLOOP(I_P))=0
 VALID(I_P,NLOOP(I_P))=1;Q1=KN(1);Q2=KN(3)
 RR0(I_P,NLOOP(I_P))=RR0(I_P,IL1)
 BURGERS(I_P,NLOOP(I_P))=BURGERS(I_P,IL1)
 TA=H*(TL(I_P,IL1,Q1)-TL(I_P,IL1,Q2));TA1=D%V(3)**TA
 !CLOSED LOOP
 N_NODE(I_P,NLOOP(I_P))=1;X=H*(PL(I_P,IL1,Q1)+PL(I_P,IL1,Q2))
 J1=1;J2=1
 DO
   DIST1=TA*TL(I_P,IL1,Q1+J1);DIST2=G*TA*TL(I_P,IL1,Q2-J2)
   IF (DIST1>0) THEN;J1=J1+1;DOT1=DIST1;END IF
   IF (DIST2>0) THEN;J2=J2+1;DOT2=DIST2;END IF
   IF ((DIST1<0).AND.(DIST2<0)) EXIT
 END DO
 IF (DIST2<-1.0D-3) THEN
   CALL PLOT(IL1,Q2-J2,-DIST2/(DOT2-DIST2))
   PL(I_P,NLOOP(I_P),N_NODE(I_P,NLOOP(I_P)))=R_PLOT
 ELSE
   PL(I_P,NLOOP(I_P),N_NODE(I_P,NLOOP(I_P)))=PL(I_P,IL1,Q2-J2)
   J2=J2+1
 END IF
 TL(I_P,NLOOP(I_P),N_NODE(I_P,NLOOP(I_P)))=TA1
 PL(I_P,NLOOP(I_P),2)=X+CU1*DIS*TA-CU2*DIS1*TA1
 N_NODE(I_P,NLOOP(I_P))=N_NODE(I_P,NLOOP(I_P))+1
 TL(I_P,NLOOP(I_P),2)=G*TA
 PL(I_P,NLOOP(I_P),3)=X+CU1*DIS*TA+CU2*DIS1*TA1
 N_NODE(I_P,NLOOP(I_P))=N_NODE(I_P,NLOOP(I_P))+1
 TL(I_P,NLOOP(I_P),3)=TA
 IF (DIST1<-1.0D-3) THEN
   CALL PLOT(IL1,Q1+J1-1,DOT1/(DOT1-DIST1))
   PL(I_P,NLOOP(I_P),4)=R_PLOT
 ELSE
   PL(I_P,NLOOP(I_P),4)=PL(I_P,IL1,Q1+J1);J1=J1+1
 END IF
 TL(I_P,NLOOP(I_P),4)=TA1
 N_NODE(I_P,NLOOP(I_P))=N_NODE(I_P,NLOOP(I_P))+1
 DO I=Q1+J1,Q2-J2
   N_NODE(I_P,NLOOP(I_P))=N_NODE(I_P,NLOOP(I_P))+1
   PL(I_P,NLOOP(I_P),N_NODE(I_P,NLOOP(I_P)))=PL(I_P,IL1,I)
   CUSPTEM(I_P,NLOOP(I_P),N_NODE(I_P,NLOOP(I_P)))=CUSP(I_P,IL1,I)
   TL(I_P,NLOOP(I_P),N_NODE(I_P,NLOOP(I_P)))=TL(I_P,IL1,I)
 END DO
 CUSPTEM(I_P,NLOOP(I_P),2)=-1;CUSPTEM(I_P,NLOOP(I_P),3)=1
 !OPEN LOOP
 J1=1;J2=1
 DO
   DIST1=TA*TL(I_P,IL1,Q1-J1);DIST2=G*TA*TL(I_P,IL1,Q2+J2)
   IF (DIST1>0) THEN;J1=J1+1;DOT1=DIST1;END IF
   IF (DIST2>0) THEN;J2=J2+1;DOT2=DIST2;END IF
   IF(((DIST1<0).OR.(Q1-J1==0)).AND.((DIST2<0).OR.
+  (Q2+J2==N_NODE(I_P,IL2)+1))) EXIT
 END DO
 UU=-DIST1/(DOT1-DIST1)
 IF (DIST1<-1.0D-1) THEN;CALL PLOT(IL1,Q1-J1,UU)
   PL(I_P,IL1,Q1-J1+1)=R_PLOT
   CURTEM(I_P,IL1,Q1-J1+1)=1.0D0/(1.0D0/CUR(I_P,IL1,Q1-J1)*(1-UU)+
+                  1.0D0/CUR(I_P,IL1,Q1-J1+1)*UU)
 ELSE IF (DIST1<0) THEN;J1=J1+1
 ELSE;J1=J1-1;END IF
 TL(I_P,IL1,Q1-J1+1)=G*TA1
 PL(I_P,IL1,Q1-J1+2)=X-CU1*DIS*TA+CU2*DIS1*TA1
 TL(I_P,IL1,Q1-J1+2)=TA
 PL(I_P,IL1,Q1-J1+3)=X-CU1*DIS*TA-CU2*DIS1*TA1
 TL(I_P,IL1,Q1-J1+3)=G*TA
 UU=DOT2/(DOT2-DIST2)
 IF (DIST2<-1.0D-1) THEN
   CALL PLOT(IL1,Q2+J2-1,UU)
   PL(I_P,IL1,Q1-J1+4)=R_PLOT
   CURTEM(I_P,IL1,Q1-J1+4)=1.0D0/(1.0D0/CUR(I_P,IL1,Q2+J2-1)
+                          *(1-UU)+1.0D0/CUR(I_P,IL1,Q2+J2)*UU)
 ELSE
   PL(I_P,IL1,Q1-J1+4)=PL(I_P,IL1,Q2+J2)
   CURTEM(I_P,IL1,Q1-J1+4)=CUR(I_P,IL1,Q2+J2);J2=J2+1
 END IF
 TL(I_P,IL1,Q1-J1+4)=G*TA1
 DO I=Q2+J2,N_NODE(I_P,IL1)
   PL(I_P,IL1,Q1+I-Q2-J1-J2+5)=PL(I_P,IL1,I)
 END DO
 DO I=Q2+J2,N_NODE(I_P,IL1)
   CUSPTEM(I_P,IL1,Q1+I-Q2-J1-J2+5)=CUSP(I_P,IL1,I)
   CURTEM(I_P,IL1,Q1+I-Q2-J1-J2+5)=CUR(I_P,IL1,I)
 END DO
 DO I=Q2+J2,N_NODE(I_P,IL1)
   TL(I_P,IL1,Q1+I-Q2-J1-J2+5)=TL(I_P,IL1,I)
 END DO
 N_NODE(I_P,IL1)=Q1+N_NODE(I_P,IL1)-Q2-J1-J2+5
 CUSPTEM(I_P,IL1,Q1-J1+2)=-1
 CUSPTEM(I_P,IL1,Q1-J1+3)=1;CUSP=CUSPTEM
 CURTEM(I_P,IL1,Q1-J1+2)=CUR(I_P,IL1,Q1)/TRA
 CURTEM(I_P,IL1,Q1-J1+3)=CUR(I_P,IL1,Q2)/TRA
 ADJ=(1/CUR(I_P,IL1,Q1)*TRA-CUR1/DIS)/((1+4*DIS**2)
+      /(8*DIS)-CUR1/DIS)
 DO I=2,N_NODE(I_P,IL1)-1
  CURTEM(I_P,IL1,I)=1/((1/CURTEM(I_P,IL1,I)-CUR1/DIS)/ADJ+CUR1/DIS)
 END DO
 CUR=CURTEM
 IF ((DIST1>0).OR.(DIST2>0)) THEN
   PL(I_P,IL1,3)=PL(I_P,IL1,N_NODE(I_P,IL1))
   PL(I_P,IL1,2)=H*(PL(I_P,IL1,1)+PL(I_P,IL1,3))
+            -D%V(3)**(PL(I_P,IL1,3)-PL(I_P,IL1,1))/10.0D0
   N_NODE(I_P,IL1)=3
   DO J=1,MAX_NODE;CUSP(I_P,IL1,J)=0;CUSPTEM(I_P,IL1,J)=0;END DO
 END IF
 CALL TAN_NOR(IL1);CALL TAN_NOR(NLOOP(I_P))
 CASE(2) !CLOSED-CLOSED
 VALID(I_P,IL2)=0;Q1=KN(1);Q2=KN(3)
 TA=H*(TL(I_P,IL1,Q1)-TL(I_P,IL2,Q2))
 X=H*(PL(I_P,IL1,Q1)+PL(I_P,IL2,Q2));TA1=D%V(3)**TA
 J1=1;J2=1;J3=1;J4=1
 DO
   IF (Q1-J1<1) THEN;J1=J1-N_NODE(I_P,IL1);END IF
   DIST1=TA*TL(I_P,IL1,Q1-J1)
   IF (DIST1>0) THEN;J1=J1+1;DOT1=DIST1;END IF
   IF (Q2+J2>N_NODE(I_P,IL2)) THEN;J2=J2-N_NODE(I_P,IL2);END IF
   DIST2=G*TA*TL(I_P,IL2,Q2+J2)
   IF (DIST2>0) THEN;J2=J2+1;DOT2=DIST2;END IF
   IF (Q2-J3<1) THEN;J3=J3-N_NODE(I_P,IL2);END IF
   DIST3=G*TA*TL(I_P,IL2,Q2-J3)
   IF (DIST3>0) THEN;J3=J3+1;DOT3=DIST3;END IF
   IF (Q1+J4>N_NODE(I_P,IL1)) THEN;J4=J4-N_NODE(I_P,IL1);END IF
   DIST4=TA*TL(I_P,IL1,Q1+J4)
   IF (DIST4>0) THEN;J4=J4+1;DOT4=DIST4;END IF
   IF ((DIST1<0).AND.(DIST2<0).AND.(DIST3<0).AND.(DIST4<0)) EXIT
 END DO
 I1=0;I1=I1+1
```

```
IF (DIST1<-1.0D-2) THEN
  CALL PLOT(IL1,Q1-J1,-DIST1/(DOT1-DIST1))
  PTEM(I_P,IL1,I1)=R_PLOT
ELSE
  PTEM(I_P,IL1,I1)=PL(I_P,IL1,Q1-J1);J1=J1+1
END IF
TTEM(I_P,IL1,I1)=G*TA1;I1=I1+1
PTEM(I_P,IL1,I1)=X-CU1*DIS*TA+CU2*DIS1*TA1;TTEM(I_P,IL1,I1)=TA
I1=I1+1;PTEM(I_P,IL1,I1)=X-CU1*DIS*TA-CU2*DIS1*TA1
TTEM(I_P,IL1,I1)=G*TA;I1=I1+1
IF (DIST2<-1.0D-2) THEN
  CALL PLOT(IL2,Q2+J2-1,DOT2/(DOT2-DIST2))
  PTEM(I_P,IL1,I1)=R_PLOT
ELSE
  PTEM(I_P,IL1,I1)=PL(I_P,IL2,Q2+J2);J2=J2+1
END IF
TTEM(I_P,IL1,I1)=G*TA1
IF(Q2-J3>Q2+J2) THEN;Q3=Q2-J3;ELSE;Q3=Q2-J3+N_NODE(I_P,IL2);END IF
DO I=Q2+J2,Q3
  I1=I1+1
  IF (I>N_NODE(I_P,IL2)) THEN;Q4=N_NODE(I_P,IL2);ELSE;Q4=0;END IF
  PTEM(I_P,IL1,I1)=PL(I_P,IL2,I-Q4)
  CUSPTEM(I_P,IL1,I1)=CUSP(I_P,IL2,I-Q4)
  TTEM(I_P,IL1,I1)=TL(I_P,IL2,I-Q4)
END DO
I1=I1+1
IF (DIST3<-1.0D-2) THEN
  CALL PLOT(IL2,Q2-J3,-DIST3/(DOT3-DIST3))
  PTEM(I_P,IL1,I1)=R_PLOT
ELSE
  PTEM(I_P,IL1,I1)=PL(I_P,IL2,Q2-J3);J3=J3+1
END IF
TTEM(I_P,IL1,I1)=TA1;I1=I1+1;I2=I1
PTEM(I_P,IL1,I1)=X+CU1*DIS*TA-CU2*DIS1*TA1;TTEM(I_P,IL1,I1)=G*TA
I1=I1+1;PTEM(I_P,IL1,I1)=X+CU1*DIS*TA+CU2*DIS1*TA1
TTEM(I_P,IL1,I1)=TA;I1=I1+1
IF (DIST4<-1.0D-2) THEN
  CALL PLOT(IL1,Q1+J4-1,DOT4/(DOT4-DIST4))
  PTEM(I_P,IL1,I1)=R_PLOT
ELSE
  PTEM(I_P,IL1,I1)=PL(I_P,IL1,Q1+J4);J4=J4+1
END IF
TTEM(I_P,IL1,I1)=TA1
IF(Q1-J1>Q1+J4) THEN;Q3=Q1-J1;ELSE;Q3=Q1-J1+N_NODE(I_P,IL1);END IF
DO I=Q1+J4,Q3
  I1=I1+1
  IF (I>N_NODE(I_P,IL1)) THEN;Q4=N_NODE(I_P,IL1);ELSE;Q4=0;END IF
  PTEM(I_P,IL1,I1)=PL(I_P,IL1,I-Q4)
  CUSPTEM(I_P,IL1,I1)=CUSP(I_P,IL1,I-Q4)
  TTEM(I_P,IL1,I1)=TL(I_P,IL1,I-Q4)
END DO
NTEM(I_P,IL1)=I1;PL=PTEM;TL=TTEM;N_NODE=NTEM;CUSP=CUSPTEM
CUSP(I_P,IL1,2)=-1;CUSP(I_P,IL1,3)=1;CUSP(I_P,IL1,I2)=-1
CUSP(I_P,IL1,I2+1)=1
CALL TAN_NOR(IL1)
CASE(3) !OPEN-CLOSED
IF (LOOPTYPE(I_P,IL1)==1) THEN
  L1=IL1;L2=IL2;VALID(I_P,IL2)=0;Q1=KN(1);Q2=KN(3)
ELSE
  L2=IL1;L1=IL2;VALID(I_P,IL1)=0;Q2=KN(1);Q1=KN(3)
END IF
TA=H*(TL(I_P,L1,Q1)-TL(I_P,L2,Q2));TA1=D%V(3)**TA
X=H*(PL(I_P,IL1,Q1)+PL(I_P,IL2,Q2));J1=1;J2=1;J3=1;J4=1
DO
  DIST1=TA*TL(I_P,L1,Q1-J1)
  IF(DIST1>-1.0D-5) THEN;J1=J1+1;DOT1=DIST1;END IF
  IF (Q2+J2>N_NODE(I_P,L2)) THEN;J2=J2-N_NODE(I_P,L2);END IF
  DIST2=G*TA*TL(I_P,L2,Q2+J2)
  IF (DIST2>-1.0D-5) THEN;J2=J2+1;DOT2=DIST2;END IF
  IF (Q2-J3<1) THEN;J3=J3-N_NODE(I_P,L2);END IF
  DIST3=G*TA*TL(I_P,L2,Q2-J3)
  IF (DIST3>-1.0D-5) THEN;J3=J3+1;DOT3=DIST3;ENDIF
  DIST4=TA*TL(I_P,L1,Q1+J4)
  IF (DIST4>-1.0D-5) THEN;J4=J4+1;DOT4=DIST4;END IF
  IF (((DIST1<-1.0D-5).OR.(Q1-J1==1)).AND.(DIST2<-1.0D-5).AND.
+  (DIST3<-1.0D-5).AND.((DIST4<-1.0D-5)
+   .OR.(Q1+J4==N_NODE(I_P,L1))))EXIT
 END DO
 I1=Q1-J1;
 IF (DIST1<0) THEN;I1=I1+1;CALL PLOT(L1,Q1-J1,-DIST1/(DOT1-DIST1))
 PTEM(I_P,L1,I1)=R_PLOT;TTEM(I_P,L1,I1)=G*TA1;END IF
 I1=I1+1;I2=I1;PTEM(I_P,L1,I1)=X-CU1*DIS*TA+CU2*DIS1*TA1
 TTEM(I_P,L1,I1)=TA;I1=I1+1
 PTEM(I_P,L1,I1)=X-CU1*DIS*TA-CU2*DIS1*TA1;TTEM(I_P,L1,I1)=G*TA
 I1=I1+1;CALL PLOT(L2,Q2+J2-1,DOT2/(DOT2-DIST2))
 PTEM(I_P,L1,I1)=R_PLOT;TTEM(I_P,L1,I1)=G*TA1
 IF (Q2-J3>Q2+J2) THEN;Q3=Q2-J3;ELSE;Q3=Q2-J3+N_NODE(I_P,L2);END IF
 DO I=Q2+J2,Q3
  I1=I1+1
  IF (I>N_NODE(I_P,L2)) THEN;Q4=N_NODE(I_P,L2);ELSE;Q4=0;END IF
  PTEM(I_P,L1,I1)=PL(I_P,L2,I-Q4)
  CUSPTEM(I_P,L1,I1)=CUSP(I_P,L2,I-Q4)
  TTEM(I_P,L1,I1)=TL(I_P,L2,I-Q4)
 END DO
 I1=I1+1;CALL PLOT(L2,Q2-J3,-DIST3/(DOT3-DIST3))
 PTEM(I_P,L1,I1)=R_PLOT
 TTEM(I_P,L1,I1)=TA1;I1=I1+1;I3=I1
 PTEM(I_P,L1,I1)=X+CU1*DIS*TA-CU2*DIS1*TA1;TTEM(I_P,L1,I1)=G*TA
 I1=I1+1;PTEM(I_P,L1,I1)=X+CU1*DIS*TA+CU2*DIS1*TA1
 TTEM(I_P,L1,I1)=TA
 IF(DIST4<0) THEN;I1=I1+1;CALL PLOT(L1,Q1+J4-1,DOT4/(DOT4-DIST4))
 PTEM(I_P,L1,I1)=R_PLOT;TTEM(I_P,L1,I1)=TA1;END IF
 DO I=Q1+J4,N_NODE(I_P,L1);I1=I1+1;PTEM(I_P,L1,I1)=PL(I_P,L1,I)
  CUSPTEM(I_P,L1,I1)=CUSP(I_P,L1,I);TTEM(I_P,L1,I1)=TL(I_P,L1,I)
 END DO
 NTEM(I_P,L1)=I1;PL=PTEM;TL=TTEM;N_NODE=NTEM;CUSP=CUSPTEM
 CUSP(I_P,L1,I2)=-1
 CUSP(I_P,L1,I2+1)=1;CUSP(I_P,L1,I3)=-1
 CUSP(I_P,L1,I3+1)=1;CALL TAN_NOR(L1)
 CASE(4) !OPEN-OPEN
 Q1=KN(1);Q2=KN(3);TA=H*(TL(I_P,IL1,Q1)-TL(I_P,IL2,Q2))
 TA1=D%V(3)**TA
 X=H*(PL(I_P,IL1,Q1)+PL(I_P,IL2,Q2));J1=1;J2=1
 DO
  DIST1=TA*TL(I_P,IL1,Q1-J1)
  IF (DIST1>0) THEN;J1=J1+1;DOT1=DIST1;END IF
  DIST2=G*TA*TL(I_P,IL2,Q2+J2)
  IF (DIST2>0) THEN;J2=J2+1;DOT2=DIST2;END IF
  IF(((DIST1<0).OR.(Q1-J1==1)).AND.((DIST2<0).OR.
+  (Q2+J2==N_NODE(I_P,IL2)))) EXIT
 END DO
 I1=0;I2=1;
 IF (DIST1<0) THEN;I1=I1+1;I2=2
  IF (DIST1<-1.0D-3) THEN;CALL PLOT(IL1,Q1-J1,-DIST1/(DOT1-DIST1))
  PTEM(I_P,IL1,Q1-J1+I1)=R_PLOT;ELSE;J1=J1+1;END IF
  TTEM(I_P,IL1,Q1-J1+I1)=G*TA1
 END IF
 I1=I1+1;PTEM(I_P,IL1,Q1-J1+I1)=X-CU1*DIS*TA+CU2*DIS1*TA1
 TTEM(I_P,IL1,Q1-J1+I1)=TA;I1=I1+1
 PTEM(I_P,IL1,Q1-J1+I1)=X-CU1*DIS*TA-CU2*DIS1*TA1
 TTEM(I_P,IL1,Q1-J1+I1)=G*TA
 IF (DIST2<0) THEN;I1=I1+1
```

```
  IF (DIST2<-1.0D-3) THEN
    CALL PLOT(IL2,Q2+J2-1,DOT2/(DOT2-DIST2))
    PTEM(I_P,IL1,Q1-J1+I1)=R_PLOT
  ELSE
    PTEM(I_P,IL1,Q1-J1+I1)=PL(I_P,IL2,Q2+J2)
    J2=J2+1
  END IF
  TTEM(I_P,IL1,Q1-J1+I1)=G*TA1
END IF
DO I=Q2+J2,N_NODE(I_P,IL2)
  PTEM(I_P,IL1,Q1+I-Q2-J1-J2+I1+1)=PL(I_P,IL2,I)
  CUSPTEM(I_P,IL1,Q1+I-Q2-J1-J2+I1+1)=CUSP(I_P,IL2,I)
  TTEM(I_P,IL1,Q1+I-Q2-J1-J2+I1+1)=TL(I_P,IL2,I)
END DO
NTEM(I_P,IL1)=Q1+N_NODE(I_P,IL2)-Q2-J1-J2+I1+1;
CUSPTEM(I_P,IL1,Q1-J1+I2)=-1;CUSPTEM(I_P,IL1,Q1-J1+I2+1)=1
J1=1;J2=1
DO
  DIST1=TA*TL(I_P,IL1,Q1+J1)
  IF (DIST1>0) THEN;J1=J1+1;DOT1=DIST1; END IF
  DIST2=G*TA*TL(I_P,IL2,Q2-J2)
  IF (DIST2>0) THEN;J2=J2+1;DOT2=DIST2;END IF
  IF (((DIST1<0).OR.(Q1+J1==N_NODE(I_P,IL1))).AND.((DIST2<0)
+     .OR.(Q2-J2==1))) EXIT
END DO
I1=0;I2=1
IF (DIST2<0) THEN;I1=I1+1;I2=2
  IF (DIST2<-1.0D-3) THEN;CALL PLOT(IL2,Q2-J2,-DIST2/(DOT2-DIST2))
  PTEM(I_P,IL2,Q2-J2+I1)=R_PLOT;ELSE;J2=J2+1;END IF
  TTEM(I_P,IL2,Q2-J2+I1)=TA1
END IF
I1=I1+1;PTEM(I_P,IL2,Q2-J2+I1)=X+CU1*DIS*TA-CU2*DIS1*TA1
TTEM(I_P,IL2,Q2-J2+I1)=G*TA;I1=I1+1
PTEM(I_P,IL2,Q2-J2+I1)=X+CU1*DIS*TA+CU2*DIS1*TA1
TTEM(I_P,IL2,Q2-J2+I1)=TA
IF (DIST1<0.0D0) THEN;I1=I1+1
  IF (DIST1<-1.0D-3) THEN
    CALL PLOT(IL1,Q1+J1-1,DOT1/(DOT1-DIST1))
    PTEM(I_P,IL2,Q2-J2+I1)=R_PLOT
  ELSE
    PTEM(I_P,IL2,Q2-J2+I1)=PL(I_P,IL1,Q1+J1)
    J1=J1+1
  END IF
  TTEM(I_P,IL2,Q2-J2+I1)=TA1
END IF
DO I=Q1+J1,N_NODE(I_P,IL1)
  PTEM(I_P,IL2,Q2+I-Q1-J1-J2+I1+1)=PL(I_P,IL1,I)
  CUSPTEM(I_P,IL2,Q2+I-Q1-J1-J2+I1+1)=CUSP(I_P,IL1,I)
  TTEM(I_P,IL2,Q2+I-Q1-J1-J2+I1+1)=TL(I_P,IL1,I)
END DO
NTEM(I_P,IL2)=Q2+N_NODE(I_P,IL1)-Q1-J1-J2+I1+1
CUSPTEM(I_P,IL2,Q2-J2+I2)=-1;CUSPTEM(I_P,IL2,Q2-J2+I2+1)=1
PL=PTEM;TL=TTEM;N_NODE=NTEM;CUSP=CUSPTEM
CALL TAN_NOR(IL1);CALL TAN_NOR(IL2)
END SELECT
WRITE(*,*)"ANNIHILATE",IL1,IL2
END SUBROUTINE ANNIHILATE

!************************************************************!

    SUBROUTINE ANNIHILATION
USE VECTORS;USE VARIABLES
INTEGER::JOINCASE,I,J,IS,I1
IF (IL_ANN==JL_ANN) THEN;JOINCASE=1
ELSE;JOINCASE=2+LOOPTYPE1(I_P,IL_ANN)+LOOPTYPE1(I_P,JL_ANN);END IF
SELECT CASE(JOINCASE)
```

```
CASE(1)!OPEN-ITSELF
  LOOPCHANGE(I_P,NLOOP1(I_P))=1
  NLOOP1(I_P)=NLOOP1(I_P)+1;LOOPTYPE1(I_P,NLOOP1(I_P))=0
  NSEG(I_P,NLOOP1(I_P))=NSEG(I_P,IL_ANN)
  VALSTOP(I_P,NLOOP1(I_P))=1
  DO I=1,NSEG(I_P,IL_ANN)
    PS(I_P,NLOOP1(I_P),I)=PS(I_P,IL_ANN,I)
    PE(I_P,NLOOP1(I_P),I)=PE(I_P,IL_ANN,I)
    CC(I_P,NLOOP1(I_P),I)=CC(I_P,IL_ANN,I)
    CC1(I_P,NLOOP1(I_P),I)=CC1(I_P,IL_ANN,I)
    NL1(I_P,NLOOP1(I_P),I)=NL1(I_P,IL_ANN,I)
    RR1(I_P,NLOOP1(I_P),I)=RR1(I_P,IL_ANN,I)
    VALID1(I_P,NLOOP1(I_P),I)=VALID1(I_P,IL_ANN,I)
  END DO
  PE(I_P,IL_ANN,I_ANN)=PE(I_P,JL_ANN,J_ANN)
  CALL CEN_CUR(IL_ANN,I_ANN);CALL MAX_RAD(IL_ANN,I_ANN)
  RR1(I_P,IL_ANN,I_ANN)=RMAX1
  IS=I_ANN
  DO
    IS=NL1(I_P,IL_ANN,IS);VALID1(I_P,IL_ANN,IS)=0
    IF(IS==J_ANN)EXIT
  END DO
  NL1(I_P,IL_ANN,I_ANN)=NL1(I_P,JL_ANN,J_ANN)

  IS=1
  DO
    VALID1(I_P,NLOOP1(I_P),IS)=0;IS=NL1(I_P,IL_ANN,IS)
    IF(IS==0)EXIT
  END DO
  PS(I_P,NLOOP1(I_P),1)=
+         PS(I_P,NLOOP1(I_P),NL1(I_P,NLOOP1(I_P),I_ANN))
  PE(I_P,NLOOP1(I_P),1)=
+         PE(I_P,NLOOP1(I_P),NL1(I_P,NLOOP1(I_P),I_ANN))
  CALL CEN_CUR(NLOOP1(I_P),1);CALL MAX_RAD(NLOOP1(I_P),1)
  RR1(I_P,NLOOP1(I_P),1)=RMAX1;VALID1(I_P,NLOOP1(I_P),1)=1
  NL1(I_P,NLOOP1(I_P),1)=
+         NL1(I_P,NLOOP1(I_P),NL1(I_P,NLOOP1(I_P),I_ANN))
  PE(I_P,NLOOP1(I_P),J_ANN)=PS(I_P,NLOOP1(I_P),1)
  CALL CEN_CUR(NLOOP1(I_P),J_ANN);CALL MAX_RAD(NLOOP1(I_P),J_ANN)
  RR1(I_P,NLOOP1(I_P),J_ANN)=RMAX1;VALID1(I_P,NLOOP1(I_P),J_ANN)=1
  NL1(I_P,NLOOP1(I_P),J_ANN)=1

CASE(2)
CASE(3)
CASE(4)
END SELECT
END SUBROUTINE ANNIHILATION

!**************************************************************!

   SUBROUTINE ARRANGE(ILL)
USE VECTORS;USE VARIABLES
INTEGER,INTENT(IN)::ILL
INTEGER,DIMENSION(0:MAX_NODE)::SN
TYPE(VECTOR)::MC
DOUBLE PRECISION::DD,E,RMAX,K0,CURVA,KMAX,KMIN,UU,U2,RAVE
INTEGER::I,J,N0,NN,J1,N3,J2
KMAX=0
KMIN=999
MC%V(1)=0
MC%V(2)=0
MC%V(3)=0
DO I=1,N_NODE(I_P,ILL)
  MC=MC+PL(I_P,ILL,I)
  IF (CUR(I_P,ILL,I)>KMAX) THEN
    KMAX=CUR(I_P,ILL,I)
  END IF
  IF (CUR(I_P,ILL,I)<KMIN) THEN
    KMIN=CUR(I_P,ILL,I)
  END IF
END DO
E=1.0/N_NODE(I_P,ILL)
MC=E*MC
RMAX=0
CURVA=0
RAVE=0
DO I=1,N_NODE(I_P,ILL)
  CURVA=CURVA+ABS(CUR(I_P,ILL,I))
  DD=MAG(PL(I_P,ILL,I)-MC)
  RAVE=RAVE+DD
  IF (DD>RMAX) THEN
    RMAX=DD
  END IF
END DO
NTEM(I_P,ILL)=0
K0=CURVA/N_NODE(I_P,ILL)!;K0=2/RAVE*N_NODE(I_P,ILL)!K0=3.0*1/RMAX
IF (NR(I_P)==1.0D0) THEN
  DO I=1,N_NODE(I_P,ILL)
    IF (CUR(I_P,ILL,I)>K0) THEN
      SN(I)=1
    ELSE
      SN(I)=-1
    END IF
  END DO
  IF (LOOPTYPE(I_P,ILL)==1) THEN
    SN(1)=1
    SN(N_NODE(I_P,ILL))=1
    SN(N_NODE(I_P,ILL)+1)=-1
  ELSE
    IF (ILL==1) THEN
      SN(1)=-1
      SN(N_NODE(I_P,ILL))=-1
    ELSE
      SN(1)=1
      SN(N_NODE(I_P,ILL))=1
    END IF
    SN(N_NODE(I_P,ILL)+1)=-SN(N_NODE(I_P,ILL))
  END IF
  N0=1
  NN=1
  DO I=1,N_NODE(I_P,ILL)
    IF (SN(I)*SN(I+1)>0) THEN
      NN=NN+1
    ELSE
      IF (NN==1) THEN
        IF ((LOOPTYPE(I_P,ILL)==1).AND.(I==N_NODE(I_P,ILL))) THEN
          CALL ADD_POINT(ILL,I-1,0.5D0)
        END IF
        IF (((I/=1).AND.(I/=N_NODE(I_P,ILL))).OR.(SN(I)>0)) THEN
          CALL ADD_POINT(ILL,I,0.0D0)
        END IF
        IF ((LOOPTYPE(I_P,ILL)==1).AND.(I==1)) THEN
          CALL ADD_POINT(ILL,I,0.5D0)
        END IF
      ELSE
        IF (SN(I)>0) THEN
          IF (NN<10) THEN
            DO J=0,NN
              N3=N0+INT((NN-1)*J/NN)
              UU=1.0*MOD((NN-1)*J,NN)/NN
              CALL ADD_POINT(ILL,N3,UU)
            END DO
          ELSE
            DO J=0,NN+1
              N3=N0+INT((NN-1)*J/(NN+1))
              CALL ADD_POINT(ILL,N3,1.0D0*
+               MOD((NN-1)*J,NN+1)/(NN+1))
            END DO
          END IF
        ELSE
          IF (NN>10) THEN
            DO J=1,NN-1;N3=N0-1+INT((NN+1)*J/NN)
              CALL ADD_POINT(ILL,N3,1.0D0*MOD((NN+1)*J,NN)/NN)
            END DO
          ELSE
            DO J=1,NN
              IF (CUR(I_P,ILL,N0+J-1)>=0.0D0) THEN
                CALL ADD_POINT(ILL,N0+J-1,0.0D0)
              END IF
            END DO
          END IF
        END IF
      END IF
      N0=I+1
      NN=1
    END IF
  END DO
ELSE IF (JOIN(I_P)/=1) THEN
  DO J=1,N_NODE(I_P,ILL)
    CALL ADD_POINT(ILL,J,0.0D0)
  END DO
ELSE
  DO J=1,N_NODE(I_P,ILL)
    IF (J==2) THEN
      J2=-1
    ELSE
      IF (J==N_NODE(I_P,ILL)-1) THEN
        J2=1
      ELSE
        J2=0
      END IF
    END IF
    U2=1.0D0*J2*(0.05)*LOOPTYPE(I_P,ILL)
    J2=INT(J2*(J2-1)/2)*LOOPTYPE(I_P,ILL)
    J1=INT(CUSP(I_P,ILL,J)*(CUSP(I_P,ILL,J)-1)/2)
    IF ((CUSP(I_P,ILL,J)==2).OR.(CUSP(I_P,ILL,J)==-2)) THEN
      CALL ADD_POINT(ILL,J,0.0D0)
    ELSE
      CALL ADD_POINT(ILL,J-J1-J2,J1+1.0D0*CUSP(I_P,ILL,J)
+       *(0.4+0.6*NR(I_P)*(DENSITY-1))+J2+U2)
    END IF
  END DO
END IF
END SUBROUTINE ARRANGE

!************************************************************!

   SUBROUTINE CEN_CUR(ILL,IS)
USE VECTORS;USE VARIABLES
INTEGER,INTENT(IN)::ILL,IS
DOUBLE PRECISION::EM,XX,C1,C2,C3,CA,YI,DR0
TYPE(VECTOR)::E, BLOC
E=PE(I_P,ILL,IS)-PS(I_P,ILL,IS)
BLOC=ES(I_P)*BURGERS(I_P,IL)
CA=UV(E)*UV(BLOC)
C1=(1-NU*CA**2)/2/PI/(1-NU)+NU*(2*CA**2-1)/PI/(1-NU)
C2=(21+CA**2)/32/PI+(2*CA**2-1)/PI
C3=2*SHEAR(I_P,ILL,IS)!/MU
```

```
 R0=1/C3
 DO
   YI=C3*R0-C1*DLOG(8*R0)+C2
   DR0=YI/(C3-C1/R0)
   R0=R0-DR0
   IF (DABS(DR0/R0)<1.0D-6) EXIT
 END DO
 !F(I)=C3*R0
 IF(EM/2>R0+ROBS) THEN
   CC(I_P,ILL,IS)=H*(PS(I_P,ILL,IS)+PE(I_P,ILL,IS));RMAX1=EM/2
 ELSE
   XX=DSQRT((R0+ROBS)**2-EM*EM/4)/EM;RMAX1=R0+ROBS
   CC(I_P,ILL,IS)=H*(PS(I_P,ILL,IS)+PE(I_P,ILL,IS))+XX*D%V(3)**E
 END IF
 END SUBROUTINE CEN_CUR

 !*************************************************************!

      SUBROUTINE COMPUTE_DEFORMATION
 USE VECTORS
 USE VARIABLES

 CALL EQUILIBRIUM
 STOP

 IF (CAD_PLOT=="3DPOLY") THEN
   WRITE(11,FMT=4) "3DPOLY"
4   FORMAT(A6)
 ELSE
   WRITE(11,FMT=5) "DONUT 0 2"
5   FORMAT(A9)
 END IF

   I_TIME=0
   I_TIME1=0

   DO
     IF (N_LOOP>0) THEN
       I_TIME=I_TIME+1
       CALL LOOP
     END IF
     IF (N_LOOP_OBS>0) THEN
       I_TIME1=I_TIME1+1
       CALL LOOP_OBS
     END IF
     IF ((I_TIME==N_TIMES).OR.(I_TIME1==N_TIMES)) EXIT
     WRITE(*,*) "STEPS  ",I_TIME
   END DO

 END SUBROUTINE COMPUTE_DEFORMATION

 !*************************************************************!

 SUBROUTINE COMPUTE_ELASTIC_FIELD(Q,STRAIN_Q,DISPLACEMENT,
+                         I_P2, IL2)

        USE VECTORS
        USE VARIABLES

        IMPLICIT NONE
        INTEGER:: I_P1, IL1, ID_NODE_L1, I_P2, IL2
        TYPE(MATRIX):: F1, F2
        TYPE(VECTOR):: F3

        TYPE(VECTOR),INTENT (IN)::Q                              !! THE FIELD POINT
        TYPE(VECTOR),INTENT(OUT)::DISPLACEMENT
```

```
          TYPE(MATRIX),INTENT(OUT):: STRAIN_Q

          SIG_Q=ZERO
          STRAIN_Q=ZERO
          DISPLACEMENT=ZERO%V(1)
  DO I_P1=1,N_PLANE                                    !!FOR EACH PLANE
   DO IL1=1,NLOOP(I_P1)                                !!FOR EACH LOOP ON THAT PLANE
    DO ID_NODE_L1=1,N_NODE(I_P1,IL1)                   !!FOR EACH NODE ON THAT LOOP
     !IF ((I_P1/=I_P2).OR.(IL1/=IL2)) THEN
     CALL LINE_INTEGRAL(Q,F1,F2,F3,I_P1,IL1,ID_NODE_L1)
     SIG_Q=SIG_Q+F1
     STRAIN_Q=STRAIN_Q+F2
     DISPLACEMENT=DISPLACEMENT+F3
     !ELSE
     !ENDIF
    END DO
   END DO
  END DO

  END SUBROUTINE COMPUTE_ELASTIC_FIELD

!***************************************************************!

     SUBROUTINE COMPUTE_ENERGY
USE VARIABLES
USE VECTORS

INTEGER::I_P1,IL1,ID_NODE_L1,I_P2,IL2,ID_NODE_L2
DOUBLE PRECISION::ENERGY_I, SELF_ENERGY, F4,BB,PP


     DO I_P1=1,N_PLANE
      DO IL1=1,NLOOP(I_P1)
       DO I_P2=1,N_PLANE
        DO IL2=1,NLOOP(I_P2)
         ENERGY(I_P1,I_P2,IL1,IL2)= 0.D0
        END DO
       END DO
      END DO
     END DO

     DO I_P1=1,N_PLANE
      DO IL1=1,NLOOP(I_P1)
       BB=MAG(BURGERS(I_P1,IL1))
       PP=PERIMETER(I_P1,IL1)
       DO I_P2=1,N_PLANE
         DO IL2=1,NLOOP(I_P2)
         DO ID_NODE_L1=1,N_NODE(I_P1,IL1)
          DO ID_NODE_L2=1,N_NODE(I_P2,IL2)
           CALL DOUBLE_INTEGRAL(I_P1,I_P2,IL1,IL2,
+                                    ID_NODE_L1,ID_NODE_L2, F4)
                ENERGY(I_P1,I_P2,IL1,IL2)  = F4+
+                                    ENERGY(I_P1,I_P2,IL1,IL2)
           END DO
          END DO
          IF(I_P1==I_P2.AND.IL1==IL2) THEN
          SELF_ENERGY = ENERGY(I_P1,I_P2,IL1,IL2)/2.0
          SELF_ENERGY=SELF_ENERGY/(BB*BB*PP)
         END IF
          WRITE(13,*) "ENERGY IP1,IP2,IL1,IL2  : ",
+              ENERGY(I_P1,I_P2,IL1,IL2),I_P1,I_P2,IL1,IL2
          WRITE(13,*) "SELF_ENERGY: I_P1,IL1  ",
+                self_energy,I_P1,IL1
         END DO
        END DO
      END DO
     END DO
 END SUBROUTINE COMPUTE_ENERGY

 !**************************************************************!

     SUBROUTINE COMPUTE_FORCE(Q,T_Q,KAPPA,
+         APPLIED_F,SELF_F,PK_F,PEIRELS_F,FEM_F,TOTAL_F,
+         RESULTANT_F)

 USE VECTORS
 USE VARIABLES

 TYPE(VECTOR),INTENT(IN)::Q,T_Q
 DOUBLE PRECISION,INTENT(IN)::KAPPA
 TYPE(VECTOR),INTENT(OUT)::APPLIED_F,SELF_F,PK_F,PEIRELS_F,FEM_F
 DOUBLE PRECISION::CA,C2A,E_ALPHA,E2_ALPHA,C1,C2,C3,CR1,CR2,CR3,
+THETA                                                   !!RUDY 4_4
 TYPE(VECTOR)::B_Q_L,PEIRELS_F_L,TOTAL_F, TOTAL_F_L,
+RESULTANT_F
 TYPE(MATRIX)::TRANS_MATRIX


 !Peach-Kohler



 !PK_F=(SIG_Q*B_Q)**T_Q

 PK_F%V(1)=0
 PK_F%V(2)=0
 PK_F%V(3)=0



 !Self-force                              per unit length
 IF(DABS(KAPPA) .NE. 0) THEN


   CA=T_Q*UV(B_Q)                       !!cosine alpha
   C2A=2*CA*CA-1
   E_ALPHA=B_Q*B_Q/(4*PI*(1-NU))*(1-NU*CA*CA)
   E2_ALPHA=B_Q*B_Q*NU/(2*PI*(1-NU))*C2A

   C1=(E_ALPHA+E2_ALPHA)*DLOG(2*8/MAG(B_Q)/(DABS(KAPPA)))
   C2=B_Q*B_Q*(21+CA*CA)/(64*PI)
   C3=B_Q*B_Q*C2A/(2*PI)

   SELF_F=KAPPA*(C1-C2-C3)*(UV(MILLER_Q)**T_Q)

 ELSE

   SELF_F%V(1)=0.0D0
   SELF_F%V(2)=0.0D0
   SELF_F%V(3)=0.0D0

 ENDIF

 !Applied force
 APPLIED_F=(SIG_APP*B_Q)**T_Q



 !FEM FORCE
 FEM_F = (SIG_ESTIMATED*B_Q)**T_Q
```

```
!NEED TO CONVERT VECTOR VALUES TO LOCAL COORDINATES

TRANS_MATRIX%V(3)=UV(MILLER_Q)
TRANS_MATRIX%V(1)=UV(d%v(3)**MILLER_Q)
TRANS_MATRIX%V(2)=TRANS_MATRIX%V(3)**TRANS_MATRIX%V(1)


!THE TOTAL FORCE WILL DETERMINE THE DIRECTION OF THE PEIRELS FORCE.

TOTAL_F =  SELF_F + PK_F + APPLIED_F + FEM_F



TOTAL_F_L=TRANS_MATRIX*TOTAL_F



TOTAL_F_L%V(3)=0.0

!PEIRELS FORCE CALCULATED HERE                                   !!RUDY 4_4

B_Q_L=TRANS_MATRIX*B_Q


 CR1=((MAG(TOTAL_F_L))*(MAG(B_Q_L)))
      CR2=(TOTAL_F_L*B_Q_L)
      CR3 = CR2/CR1


      THETA=DACOS(CR3)


PEIRELS_F_L=((-1E-6)*(1+ABS(SIN(THETA))))*UV(TOTAL_F_L)


PEIRELS_F=(TRANS(TRANS_MATRIX))*PEIRELS_F_L

IF (MAG(TOTAL_F_L) > MAG(PEIRELS_F)) THEN

RESULTANT_F = TOTAL_F + PEIRELS_F

ELSE

RESULTANT_F%V(1) = 0
RESULTANT_F%V(2) = 0
RESULTANT_F%V(3) = 0

ENDIF



END SUBROUTINE COMPUTE_FORCE

!**************************************************************!

SUBROUTINE COMPUTE_INTERACTION

USE VARIABLES
USE VECTORS

IMPLICIT NONE

TYPE(VECTOR)::Q, QL, NORM_Q,T_Q,T_QL, F_Q, TRACTION,
+                 ORIGIN_Q, DISPLACE_Q,APPLIED_F,SELF_F,PK_F,
+                 PEIRELS_F,TRAC, TOTAL_F, RESULTANT_F    !!RUDY 3_1
TYPE(MATRIX)::ES_Q,SIG_L_Q,STRAIN_Q,SIG_FEM
```

```
INTEGER::NODE_NUMBER, IP_RUDY, IL_RUDY,N_LOOP,GG !!RUDY 5/5 !!RUDY 2/29/00
DOUBLE PRECISION:: KAPPA, LOOP_AREA

TYPE (MATRIX)::SIG_LOOP,FEM_STRESS !!RUDY 3_3
TYPE (VECTOR) :: NEWDIF, FEM_F, ORIGINL  !!RUDY 5/5
INTEGER :: I,J,K,NUMBER_LOOP_NODES,NUMBER_FIELD_NODES
INTEGER ::UNIT_LENGTH, DIV_PER_SIDE, E_LENG, NUMBER_OF_NODES !!RUDY 5/5



! Initial values of field are zero

  SIG_Q=ZERO
  SIG_L_Q=ZERO
  STRAIN_Q=ZERO
  SIG_ESTIMATED = ZERO
  DISPLACE_Q=ZERO%V(1)

PRINT*, "FORCE TYPE:",FORCE_TYPE
PRINT*,"INTERACTION_TYPE:", INTERACTION_TYPE
PRINT*,"ELASTIC_FIELD_TYPE:",ELASTIC_FIELD_TYPE

! Set exclusions and specific cases

IF ((INTERACTION_TYPE=="DIPOLE").OR.
+    (INTERACTION_TYPE=="EQUILIBRIUM").OR.
+    (INTERACTION_TYPE=="ENERGY")) GOTO 110
!4 IS THE FIELD_INPUT.TXT FILE

IF (ELASTIC_FIELD_TYPE=="LOCAL") THEN
  READ (4,*) N_Q_POINTS, ORIGIN_Q, MILLER_Q
            ES_Q%v(3)=UV(MILLER_Q)
  ES_Q%v(1)=UV(d%v(3)**MILLER_Q)
  ES_Q%v(2)=ES_Q%v(3)**ES_Q%v(1)

ELSEIF (FORCE_TYPE =="SURFACE_TRACTION") THEN
READ(4,*) N_Q_POINTS, NODE_AREA



ELSEIF (FORCE_TYPE =="DYNAMIC_TRACTION") THEN                                  !!RUDY 5/1
READ(4,*) N_Q_POINTS, NODE_AREA
READ(8,*) N_LOOP, N_PLANE, NUMBER_OF_NODES, MILLER_Q, B_Q, ORIGINL

!N_LOOP=LOOP NUMBER
!N_PLANE=PLANE NUMBER
!NUMBER_OF_NODES=TOTAL NUMBER OF LOOP NODES
!MILLER= MILLER INDICIES OF THE SLIP PLANE
!ORIGINL=THE ORIGIN OF THE SLIP PLANE (GLOBAL COORDINATES?)



ELSEIF (DEFECT_TYPE=="SPHERICAL") THEN
READ (4,*) N_Q_POINTS,DEFECT_RADIUS

ELSEIF (DEFECT_TYPE=="SMALL_LOOP") THEN
READ (4,*) N_Q_POINTS,LOOP_AREA, LOOP_NORMAL

ELSEIF (FORCE_TYPE=="LINE_FORCE") THEN
READ (4,*) N_Q_POINTS,ORIGIN_Q, MILLER_Q,B_Q,IP_RUDY, IL_RUDY

ELSEIF (FORCE_TYPE == "DYNAMIC_FORCE") THEN
READ(8,*) N_LOOP, N_PLANE, N_Q_POINTS, MILLER_Q, B_Q, ORIGINL
```

```
      ELSE
      READ(4,*)  N_Q_POINTS, NODE_AREA
      ENDIF

      ! Read and then calculate point-by-point

      WRITE(13,*)"NODE,RESULTANT FORCE, TOTAL FORCE,PEIRELS FORCE,"
      WRITE(13,*)"APPLIED FORCE,SELF FORCE AND FEM FORCE"

      !N_Q_POINTS::NUMBER OF LOOP NODES (FOR "LINE_FORCE")
      !ORIGIN_Q::THE LOOP CENTER

       ES_Q%v(3)=UV(MILLER_Q)
       ES_Q%v(1)=UV(d%v(3)**MILLER_Q)
       ES_Q%v(2)=ES_Q%v(3)**ES_Q%v(1)


      !PRINT*, "NUMBER OF LOOP NODES:",N_Q_POINTS
      DO I=1,N_Q_POINTS

      !!N_Q_POINTS=TOTAL NUMBER OF FIELD NODES, OR THE TOTAL NUMBER OF LOOP NODES.



       SELECT CASE(INTERACTION_TYPE)


      !COMPUTE PEACH KOHLER FORCES HERE.

        CASE("FORCE")



        IF (FORCE_TYPE =="LINE_FORCE") THEN



         READ(4,*) QL,T_QL,KAPPA

         !QL IS THE LOOP NODE

         Q=(TRANS(ES_Q)*QL)+ORIGIN_Q  !!global coordinates


         T_Q=TRANS(ES_Q)*T_QL


        CALL COMPUTE_ELASTIC_FIELD(Q,STRAIN_Q,DISPLACE_Q,
     +                                     IP_RUDY, IL_RUDY)


        REWIND(UNIT=7)
        READ(7,222) NUMBER_FIELD_NODES
        READ(7,222) UNIT_LENGTH
        READ(7,222) DIV_PER_SIDE

222     FORMAT(I8)

        E_LENG = UNIT_LENGTH / DIV_PER_SIDE

      !PRINT*,"NUMBER OF FIELD NODES:", NUMBER_FIELD_NODES

       DO J = 1, NUMBER_FIELD_NODES

                      !UNIT=7 :: COORD_AND_FEM_STRESS_INPUT.TXT
```

```
                         READ(7,223)NODE_NUMBER,FEM_NODE,SIG_FEM%V(1)%V(1),
     +          SIG_FEM%V(1)%V(2),
     +          SIG_FEM%V(1)%V(3),SIG_FEM%V(2)%V(1),SIG_FEM%V(2)%V(2),
     +          SIG_FEM%V(2)%V(3),SIG_FEM%V(3)%V(1),SIG_FEM%V(3)%V(2),
     +          SIG_FEM%V(3)%V(3)

223                      FORMAT(I5,1X,12(ES12.5,1X))



        CALL SEEKER(SIG_FEM,NODE_NUMBER,Q,NEWDIF,E_LENG,
     +          CAPSULE,CAPSULE_STRESS,J,NUMBER_FIELD_NODES)


       END DO

             PRINT*,"SEEKER DONE",Q
             PRINT*,"ELEMENT LENGTH:",E_LENG
             DO GG=1,8
             PRINT*,CAPSULE(GG)
             ENDDO


              CALL ARRANGER(CAPSULE,CAPSULE_STRESS,E_LENG)



              CALL LOOP_NODE_STRESS_ESTIMATOR(CAPSULE,
     +           CAPSULE_STRESS,E_LENG, Q)



              !!SIG_ESTIMATED IS DIMENSIONLESS WHEN IT COMES OUT OF
              !LOOP_NODE_STRESS_ESTIMATOR

              !Q: THE LOOP NODE
              !FEM_NODE: THE FIELD NODE DEFINED BY THE FEM MODEL
              !SIG_FEM: THE STRESS ON THE FEM FIELD NODES
              !SIG_ESTIMATED: THE STRESS ESTIMATE ON THE LOOP NODES
              !FEM_F: THE FORCE ON THE LOOP NODE CALCULATED FROM THE
              !FEM STRESS


        CALL COMPUTE_FORCE(Q,T_Q,KAPPA,
     +      APPLIED_F,SELF_F,PK_F,PEIRELS_F,FEM_F, TOTAL_F, RESULTANT_F)




              !!APPLIED FORCE DEPENDS ON SIG_APP
              APPLIED_F  = MU*LATTICE*APPLIED_F     !! N/m GLOBAL COORD.RUDY 3_3
         SELF_F     = MU*LATTICE*SELF_F
              PK_F       = MU*LATTICE*PK_F
              PEIRELS_F  = MU*LATTICE*PEIRELS_F
              FEM_F      = MU*LATTICE*FEM_F
              TOTAL_F    = MU*LATTICE*TOTAL_F
              RESULTANT_F= MU*LATTICE*RESULTANT_F

      !PRINT*, "MAG OF RES AND PEIAND TOT",MAG(RESULTANT_F),
      !+MAG(PEIRELS_F),MAG(TOTAL_F)

        WRITE(13,103) Q,RESULTANT_F,TOTAL_F,PEIRELS_F,APPLIED_F,
     + SELF_F,FEM_F
```

```
          WRITE(20,101)Q, RESULTANT_F
          WRITE(21,101)Q, FEM_F
          WRITE(22,101)Q, PEIRELS_F
          WRITE(23,101)Q, APPLIED_F
          WRITE(24,101)Q, SELF_F

   !APPLIED FORCE:FROM THE APPLIED STRESS
   !SELF FORCE: FROM THE LINE TENSION
   !PK_FORCE: THE RESULT OF ELASTIC STRESS FROM OTHER LOOPS
   !PEIRELS_FORCE:THE RESISTANCE OF THE LATTICE TO DISPLACEMENT
   !BY THE LOOP


           !   ES_Q*SELF_F   !! N/m LOCAL COORDINATES



 !************************************************************!

          ELSEIF (FORCE_TYPE == "DYNAMIC_FORCE") THEN

          PRINT*, "IN DYNAMIC FORCE"

           KAPPA=1/2000

           !8 IS THE DYNAMIC_LOOP_DATA FILE

           READ(8,*) QL

           READ(8,*) T_QL


    Q=TRANS(ES_Q)*QL+ORIGINL  !!global coordinates

    T_Q= UV(TRANS(ES_Q)*T_QL)

    !PRINT*,"POINT AND TANGENT",Q, T_Q

   REWIND(UNIT=7)                          !UNIT=7 :: COORD_AND_FEM_STRESS_INPUT.TXT
   READ(7,222) NUMBER_FIELD_NODES
   READ(7,222) UNIT_LENGTH
   READ(7,222) DIV_PER_SIDE


   E_LENG = UNIT_LENGTH / DIV_PER_SIDE


  DO J = 1, NUMBER_FIELD_NODES





                    READ(7,223)NODE_NUMBER,FEM_NODE,SIG_FEM%V(1)%V(1),
+         SIG_FEM%V(1)%V(2),
+         SIG_FEM%V(1)%V(3),SIG_FEM%V(2)%V(1),SIG_FEM%V(2)%V(2),
+         SIG_FEM%V(2)%V(3),SIG_FEM%V(3)%V(1),SIG_FEM%V(3)%V(2),
+         SIG_FEM%V(3)%V(3)





   CALL SEEKER(SIG_FEM,NODE_NUMBER,Q,NEWDIF,E_LENG,
+           CAPSULE,CAPSULE_STRESS,J,NUMBER_FIELD_NODES)
```

```
  END DO


             CALL ARRANGER(CAPSULE,CAPSULE_STRESS,E_LENG)


             CALL LOOP_NODE_STRESS_ESTIMATOR(CAPSULE,
+                CAPSULE_STRESS,E_LENG, Q)



    CALL COMPUTE_FORCE(Q,T_Q,KAPPA,
+    APPLIED_F,SELF_F,PK_F,PEIRELS_F,FEM_F, TOTAL_F, RESULTANT_F)


             !!APPLIED FORCE DEPENDS ON SIG_APP
             APPLIED_F  = MU*LATTICE*APPLIED_F     !! N/m GLOBAL COORD.RUDY 3_3

             PEIRELS_F  = MU*LATTICE*PEIRELS_F
             FEM_F      = MU*LATTICE*FEM_F
             TOTAL_F    = MU*LATTICE*TOTAL_F
             RESULTANT_F= MU*LATTICE*RESULTANT_F



   WRITE(13,103) Q,RESULTANT_F,TOTAL_F,PEIRELS_F,APPLIED_F,
+   SELF_F,FEM_F

            WRITE(20,101)Q, RESULTANT_F
            WRITE(21,101)Q, FEM_F
            WRITE(22,101)Q, PEIRELS_F
            WRITE(23,101)Q, APPLIED_F
            WRITE(24,101)Q, SELF_F

 !**************************************************************!


 !COMPUTE TRACTIONS AND ANSYS INPUT HERE.

   ELSEIF(FORCE_TYPE =="SURFACE_TRACTION") THEN

    READ(4,*) NODE_NUMBER, Q, NORM_Q



    CALL COMPUTE_ELASTIC_FIELD(Q,STRAIN_Q,DISPLACE_Q,
+                     IP_RUDY, IL_RUDY)

    TRACTION=MU*G*SIG_Q*NORM_Q           !!RUDY 3_1  G=-1

    TRAC=NODE_AREA*TRACTION     !! Rudy 3_1
    !VALUE IN NEWTONS
    WRITE(14,*)"FLST,2,1,1,ORDE,1"
    WRITE(14,*)"FITEM,2,", NODE_NUMBER
    WRITE(14,*)"F,P51X,FX,", TRAC%v(1)
    WRITE(14,*)"FLST,2,1,1,ORDE,1"
    WRITE(14,*)"FITEM,2,", NODE_NUMBER
    WRITE(14,*)"F,P51X,FY,", TRAC%v(2)
    WRITE(14,*)"FLST,2,1,1,ORDE,1"
    WRITE(14,*)"FITEM,2,", NODE_NUMBER
    WRITE(14,*)"F,P51X,FZ,", TRAC%v(3)

             TRACTION=G*TRACTION
    WRITE(15,101) Q, TRACTION
!**************************************************************!
!**************************************************************!

            ELSEIF(FORCE_TYPE =="DYNAMIC_TRACTION") THEN                    !!RUDY 6/1


            DO K=1,NUMBER_OF_NODES

            READ(8,*) PL(1,1,N_NODE(1,1))

            READ(8,*) TL(1,1,N_NODE(1,1))

      ENDDO

      REWIND (UNIT=8)

      READ(4,*) NODE_NUMBER, Q, NORM_Q

      IP_RUDY=1
      IL_RUDY=1

    CALL COMPUTE_ELASTIC_FIELD(Q,STRAIN_Q,DISPLACE_Q,
+                         IP_RUDY, IL_RUDY)   !!Q IS THE FIELD POINT



      TRACTION=MU*G*SIG_Q*NORM_Q           !!RUDY 3_1  G=-1

      TRAC=NODE_AREA*TRACTION     !! Rudy 3_1
      !VALUE IN NEWTONS
      WRITE(14,*)"FLST,2,1,1,ORDE,1"
      WRITE(14,*)"FITEM,2,", NODE_NUMBER
      WRITE(14,*)"F,P51X,FX,", TRAC%v(1)
      WRITE(14,*)"FLST,2,1,1,ORDE,1"
      WRITE(14,*)"FITEM,2,", NODE_NUMBER
      WRITE(14,*)"F,P51X,FY,", TRAC%v(2)
      WRITE(14,*)"FLST,2,1,1,ORDE,1"
      WRITE(14,*)"FITEM,2,", NODE_NUMBER
      WRITE(14,*)"F,P51X,FZ,", TRAC%v(3)

      WRITE(15,101) Q, TRAC   !! REVERSED TRACTIONS UNITS=N



    ENDIF

!**************************************************************!
!**************************************************************!

! ELASTIC FIELD VALUES ARE COMPUTED HERE.

    CASE("ELASTIC_FIELD")

      IF (ELASTIC_FIELD_TYPE=="GLOBAL") THEN

      READ(4,*) NODE_NUMBER,Q


      CALL COMPUTE_ELASTIC_FIELD(Q,STRAIN_Q,DISPLACE_Q,
+                        IP_RUDY, IL_RUDY)

      WRITE(15,102) Q,SIG_Q%V(1)%V(1),SIG_Q%V(1)%V(2),
+                                   SIG_Q%V(1)%V(3),SIG_Q%V(2)%V(2),
+                  SIG_Q%V(2)%V(3),SIG_Q%V(3)%V(3),DISPLACE_Q

     ELSEIF(ELASTIC_FIELD_TYPE=="LOCAL") THEN

      READ(4,*) QL
      Q=(TRANS(ES_Q)*QL)+ORIGIN_Q
      CALL COMPUTE_ELASTIC_FIELD(Q,STRAIN_Q,DISPLACE_Q,
+                        IP_RUDY, IL_RUDY)
      !WRITE(*,*)"Q SIG_Q  ",Q, SIG_Q
      SIG_L_Q=ES_Q*SIG_Q*TRANS(ES_Q)
      WRITE(15,102) QL%V(1),QL%V(2),SIG_L_Q%V(1),SIG_L_Q%V(2)%V(2)
+                 ,
+                 SIG_L_Q%V(2)%V(3),SIG_L_Q%V(3)%V(3),DISPLACE_Q

     ENDIF


     CASE("DEFECT")
     READ(4,*) Q
     CALL COMPUTE_ELASTIC_FIELD(Q,STRAIN_Q,DISPLACE_Q,
+                        IP_RUDY, IL_RUDY)

     IF (DEFECT_TYPE=="SPHERICAL") THEN
         !Calculate energy
     WRITE(13,102) Q!,energy
      ELSEIF (DEFECT_TYPE=="SMALL_LOOP") THEN
         !Calculate energy, moment & force
     WRITE(13,102) Q!,energy, moment and force
     ENDIF

     CASE("KMC")

          CASE ("TETRAHEDRON")

    END SELECT
   END DO

 110 CONTINUE

    !IF (FORCE_TYPE == 'LINE_FORCE') THEN
    ! DO I=1,N_Q_POINTS                                  !JIANMING 4_14
    !  READ(4,*)ELEMENT(I,1),ELEMENT(I,2)
    ! ENDDO



    ! I_P=IP_RUDY;IL=IL_RUDY
    ! N_NODE(I_P,IL)=N_Q_POINTS



     !CALL DEFORMATION


     !DO I=1,N_Q_POINTS

     ! WRITE(16,*) PL(I_P,IL,I)%V(1),PL(I_P,IL,I)%V(2)

     !ENDDO

    !ENDIF


   SELECT CASE(INTERACTION_TYPE)
    CASE("ENERGY")
     CALL COMPUTE_ENERGY
    CASE("SINGLE_EQUILIBRIUM")
      CALL SINGLE_EQUILIBRIUM(PL(1,1,1),PL(1,1,3))
    CASE("DIPOLE")
      CALL DIPOLE
   END SELECT

101      FORMAT(6(2X,E12.6))
102      FORMAT(11(2X,E12.6))
103      FORMAT(24(E12.6,2X))
   END SUBROUTINE COMPUTE_INTERACTION

   !*************************************************************!

        SUBROUTINE CURVATURE(UT,UN)
   USE VECTORS;USE VARIABLES
   TYPE(VECTOR),INTENT(IN)::UT,UN
   DOUBLE PRECISION::C1,C2,C3,CA,YI,DR0,F_NORM
   TYPE(VECTOR)::BLOC,UTG,UNG,F_TOT
   BLOC=ES(I_P)*BURGERS(I_P,IL)
!    CA=UT*UV(BURGERS(I_P,IL))
   CA=UT*UV(BLOC)
   C1=(1-NU*CA**2)/4/PI/(1-NU)+NU*(2*CA**2-1)/2/PI/(1-NU)
   C2=(21+CA**2)/64/PI+(2*CA**2-1)/2/PI

   UTG=TRANS(ES(I_P))*UT
   UNG=TRANS(ES(I_P))*UN
   F_TOT=((SIG_APP+SIG)*BURGERS(I_P,IL))**UTG
   F_NORM=F_TOT*UNG

   IF (CASE_TYPE=="DIPOLE") THEN
    CALL DISPLACE_NODES(QG,UT,UN,F_NORM)
    F_NORM=F_NORM/MS
   END IF

   C3=DABS(F_NORM)/MAG(BURGERS(I_P,IL))
   IF (C3<0.1/MAG(PL(I_P,IL,3)-PL(I_P,IL,1))) THEN
    C3=0.1/MAG(PL(I_P,IL,3)-PL(I_P,IL,1))
   END IF
   R0=1/C3
   DO
    YI=C3*R0-C1*DLOG(8*R0)+C2
    DR0=YI/(C3-C1/R0)
    R0=R0-DR0
    IF (DABS(DR0/R0)<1.0D-9) EXIT
   END DO
   END SUBROUTINE CURVATURE

   !*************************************************************!

        SUBROUTINE DATA_READER
   USE VECTORS
   USE VARIABLES

   NAMELIST /MATERIAL/MU, NU, LATTICE, TEMP, MS, A_CUBE, APPLIED_SIG,
  +                CRIT,CALCULATION_TYPE, INTERACTION_TYPE,
  +                SEG_TYPE,RMIN,CAD_PLOT, INTERVAL,
  +                ELASTIC_FIELD_TYPE,FORCE_TYPE,DEFECT_TYPE
  +                              , MOBILITY,DTIME,RTOL,ATOLL
   NAMELIST /DIMENSIONS/MAX_QUAD, MAX_LOOP, MAX_NODE, MAX_PLANE,OMAX
  +                 , N_TIMES



   READ(1,MATERIAL)
   READ(1,DIMENSIONS)
```

```
IF (CALCULATION_TYPE=="INTERACTION") THEN

CALL INTERACTION_GEOM_WRITER(FORCE_TYPE)     !!RUDY 3_9

END IF



SIG_APP%V(1)%V(1) =  APPLIED_SIG
SIG_APP           =  SIG_APP/MU           !!UNITLESS

! Allocate dimensions

CALL ALLOCATE_DIMENSIONS

REWIND(2)

READ(2,*) N_PLANE

N_LOOP=0
N_LOOP_OBS=0

DO I_P=1,N_PLANE
  READ(2,*) NLOOP(I_P),MILLER(I_P),ORIGIN(I_P),NLOOP1(I_P)

  N_LOOP=N_LOOP+NLOOP(I_P)
  N_LOOP_OBS=N_LOOP_OBS+NLOOP1(I_P)
  DO IL=1,NLOOP(I_P)
    PERIMETER(I_P,IL)=0.0D0
    READ(2,*) BURGERS(I_P,IL),N_NODE(I_P,IL),LOOPTYPE(I_P,IL)
    DO I=1,N_NODE(I_P,IL)
      READ(2,*) PL(I_P,IL,I)
    END DO
    IF (LOOPTYPE(I_P,IL)==0) THEN
      PL(I_P,IL,0)=PL(I_P,IL,N_NODE(I_P,IL))
      PL(I_P,IL,N_NODE(I_P,IL)+1)=PL(I_P,IL,1)
    END IF
    IF (N_NODE(I_P,IL)>10) THEN
      PERIMETER(I_P,IL)=PI*
+            MAG(PL(I_P,IL,1)-PL(I_P,IL,INT(N_NODE(I_P,IL)/2)+1))
    WRITE(*,*)PI
      WRITE(*,*)"PERIMETER(I_P,IL)  I_P,IL",PERIMETER(I_P,IL)
    ELSE
      DO I=1,N_NODE(I_P,IL)-LOOPTYPE(I_P,IL)
        PERIMETER(I_P,IL)=PERIMETER(I_P,IL)+
+                   MAG(PL(I_P,IL,I+1)-PL(I_P,IL,I))
      END DO
    END IF
  END DO

  DO IL=1,NLOOP1(I_P)
    READ(2,*) BURGERS_OB(I_P,IL),PS(I_P,IL,1),PE(I_P,IL,1)
  END DO
      !READ THE COORDINATES OF OBSTACLES
  IF (NLOOP1(I_P)>0) THEN
    READ(6,*) NOBS(I_P)
    DO I=1,NOBS(I_P)
      READ(6,*) OBS(I_P,I)
    END DO
  END IF
END DO

END SUBROUTINE DATA_READER

!*************************************************************!
SUBROUTINE DEFAULTS
```

```
 USE VARIABLES
 USE VECTORS
 IMPLICIT NONE

         PI                    =  2.*DASIN(1.D0)
          MU                         =  50D9
         NU                          =  0.3
         LATTICE   =  2.85D-10  !meters
         TEMP                        =  300D0
         MS                          =  5D4
         A_CUBE    =  1000
         SIG_APP   =  ZERO
         CRIT                        =  170
         INTERACTION_TYPE  =  "ENERGY"
         CALCULATION_TYPE  =  "INTERACTION"
         SEG_TYPE                  =  "CUBIC"

 MAX_QUAD       =  16
 MAX_LOOP       =  10
 MAX_NODE       =  30
 MAX_PLANE      =  10
 NPLOT=10
 OMAX                            =  2500
 N_TIMES                         =  100
 ROBS             =  5 !The radius of each obatacle
 ARB             =  D%V(3)
 INTERVAL         =  0.5D0
 INTERVAL2        =  INTERVAL*INTERVAL
 RMIN             =  DSQRT(2.0D0)/2.d0
 NEAREST_DIST      =  100
 RTOL                            =1.D-4
 ATOLL                           =1.D-4
 DTIME            =1.D-12
 MOBILITY         =1.D-5

END SUBROUTINE DEFAULTS
!***************************************************************!

    SUBROUTINE DIPOLE
USE VARIABLES
USE VECTORS
DOUBLE PRECISION::CT,ST,DT,LL,PLL,RATIO,PDANG,NDANG,DANG1
DOUBLE PRECISION,DIMENSION(2)::NDANG1
TYPE(VECTOR)::DANG,T0
TYPE(VECTOR),DIMENSION(2)::NORV,NORV1 !Changed to 2D array
TYPE(VECTOR),DIMENSION(2,360)::Q !Changed to 3Darray
INTEGER::I,J,J1
DT=PI/360
IL=1
DO I_P=1,2
  NQ(I_P)=180
  DO J=1,NQ(I_P)
    QD(I_P,J)=PL(I_P,IL,1)+(J/180D0)*(PL(I_P,IL,3)-PL(I_P,IL,1))
    QT(I_P,J)=PL(I_P,IL,3)-PL(I_P,IL,1)
  END DO
END DO
DO I_P=1,2
  NORV(I_P)=G*UV(D%V(3)**(PL(I_P,IL,2)-PL(I_P,IL,1)))
END DO
DO J1=1,3
  DO I_P=1,2
    RATIO=1D0
    PDANG=0.0D0
    DO
     NORV1(I_P)=NORV(I_P)
```

```
      I=1
      LL=0
      DO
        PLL=LL
        QG=TRANS(ES(I_P))*QD(I_P,I)+ORIGIN(I_P)
        CALL CURVATURE(D%V(3)**NORV(I_P),G*NORV(I_P))
        QD(I_P,I+1)=QD(I_P,I)-R0*NORV(I_P)+DT**(R0*NORV(I_P))
        QT(I_P,I+1)=D%V(3)**(DT**(R0*NORV(I_P)))
        NORV(I_P)=DT**NORV(I_P)
        I=I+1
        LL=MAG(QD(I_P,I)-QD(I_P,1))
        IF ((LL>MAG(PL(I_P,IL,3)-QD(I_P,1))).OR.(LL<PLL)) EXIT
        !INSTABILITY CONDITION LL<PLL
      END DO
      DANG=UV(QD(I_P,I)-QD(I_P,1))**UV(PL(I_P,IL,3)-QD(I_P,1))
      DANG1=UV(QD(I_P,I)-QD(I_P,1))*UV(PL(I_P,IL,3)-QD(I_P,1))
      NDANG=DACOS(DANG1)*DANG%V(3)/ABS(DANG%V(3))
      IF (PDANG==0.0D0) THEN
        RATIO=1
      ELSE
        RATIO=((NDANG-PDANG)/(PDANG/RATIO))
      END IF
      NORV(I_P)=(NDANG/RATIO)**NORV1(I_P)
      PDANG=NDANG
      !WRITE(*,*)"DANG1  DANG%V(3)",DANG1,DANG%V(3)
      IF ((MAG(DANG)<0.05D0).AND.(DANG1>0)) EXIT
    END DO
    NDANG1(I_P)=NDANG
    NQ(I_P)=I
    WRITE(*,*)"I_P  ",I_P
  END DO
END DO

!WRITE(*,*)"LINE 50"
!IF (LL<MAG(QD(I_P,NQ(I_P))-QD(I_P,1))) THEN
!  WRITE(*,*)"(2)LOOP IS UNSTABLE."
!END IF


WRITE(6,FMT=16) "3DPOLY"
16 FORMAT(A6)

DO I_P=1,2
  DO J=1,NQ(I_P)
    QG=TRANS(ES(I_P))*QD(I_P,J)+ORIGIN(I_P)
    IF ((I_P/=1).AND.(J==1)) THEN
      WRITE(11,FMT=18) "  ",QG%V(1),",",QG%V(2),",",QG%V(3)
18      FORMAT(A2,E12.6,A1,E12.6,A1,E12.6)
    ELSE
      WRITE(11,FMT=17) QG%V(1),",",QG%V(2),",",QG%V(3)
17      FORMAT(E12.6,A1,E12.6,A1,E12.6)
    END IF
  END DO
END DO

END SUBROUTINE DIPOLE

!***************************************************************!

      SUBROUTINE DISPLACE_NODES(Q,UT,UN,DELTA_P)
USE VECTORS;USE VARIABLES
TYPE(VECTOR),INTENT(IN)::Q,UT,UN
DOUBLE PRECISION,INTENT(OUT)::DELTA_P
TYPE(MATRIX)::F1,F2
TYPE(VECTOR)::F3,F_TOT, BLOC, UTG,UNG,FORCE1,FORCE2
DOUBLE PRECISION::F_NORM
```

```
  INTEGER::IP_LOC, IL_LOC
  SIG=ZERO

  DO IP_LOC=1,N_PLANE
   DO IL_LOC=1,NLOOP(IP_LOC)
     IF ((IP_LOC/=I_P).OR.(IL_LOC/=IL)) THEN
!        WRITE(*,*)"N_NODE(IP_LOC,IL_LOC)  ",N_NODE(IP_LOC,IL_LOC)
       DO ID_NODE_L=1,N_NODE(IP_LOC,IL_LOC)
         CALL LINE_INTEGRAL(Q,F1,F2,F3,IP_LOC,IL_LOC,ID_NODE_L)
         SIG=SIG+F1
!          WRITE(*,*)"IP_LOC,IL_LOC,ID_NODE_L,F1  ",
! +        IP_LOC,IL_LOC,ID_NODE_L,F1
       END DO
     END IF
   END DO
  END DO
  UTG=TRANS(ES(I_P))*UT
  UNG=TRANS(ES(I_P))*UN
  !BLOC=ES(I_P)*BURGERS(I_P,IL)
  !SIG=ZERO

  FORCE1=(SIG*BURGERS(I_P,IL))**UTG
  FORCE2=(SIG_APP*BURGERS(I_P,IL))**UTG
  !WRITE(*,*)"FORCE1 FORCE2   ",FORCE1*UNG,FORCE2*UNG

  IF (DABS(FORCE1*UNG)>DABS(FORCE2*UNG)) THEN
   F_TOT=ZERO%V(1)
  ELSE
   F_TOT=FORCE1+FORCE2
  END IF
!  WRITE(*,*)"SIG  ",SIG
  !SIG=ES(I_P)*SIG*TRANS(ES(I_P))

  F_NORM=F_TOT*UNG
  DELTA_P=MS*F_NORM
!   WRITE(*,*)"DELTA_P   ",DELTA_P
  END SUBROUTINE DISPLACE_NODES

  !**************************************************************!

       SUBROUTINE DISPLACEMENT(ILL,I,UU)
  USE VECTORS;USE VARIABLES
  INTEGER,INTENT(IN)::ILL,I;DOUBLE PRECISION,INTENT(IN)::UU
  DOUBLE PRECISION::DD,DD1,CURVA,U4,U3,U2,DIS2,CO, DELTA_P, CALPHA,
 +                INCREMENT
  TYPE(VECTOR)::TP1,TTP,NN, QL,Q
  DOUBLE PRECISION, EXTERNAL::SELF_DELTA
  U2=UU*UU;U3=U2*UU;U4=U3*UU !QUINTIC
  CALL TANG(ILL,I,UU)
  TP1=60*(2*U3-3*U2+UU)*(PL(I_P,ILL,I+1)-PL(I_P,ILL,I))+
 +   (-60*U3+96*U2-36*UU)*T2(I_P,ILL,I)*TL(I_P,ILL,I)+
 +   (-60*U3+84*U2-24*UU)*T1(I_P,ILL,I+1)*TL(I_P,ILL,I+1)+
 +   (-10*U3+18*U2-9*UU+1)*N2(I_P,ILL,I)*NL(I_P,ILL,I)+
 +   (10*U3-12*U2+3*UU)*N1(I_P,ILL,I+1)*NL(I_P,ILL,I+1)
  DD1=MAG(RT);TTP=RT**TP1;NN=TTP**RT;DD=MAG(NN);NN=NN/DD
  TTEM(I_P,ILL,NTEM(I_P,ILL))=UV(RT);NT(I_P,ILL,NTEM(I_P,ILL))=NN
  !IF (I==N_NODE(I_P,ILL)) THEN
  !WRITE(*,*)"TL",TL(I_P,ILL,I+1)%V(1),TL(I_P,ILL,I+1)%V(2);END IF
  IF (TTP%V(3)<0.0D0) THEN
   NN=G*NN;NT(I_P,ILL,NTEM(I_P,ILL))=NN
  END IF
  !CURVA=MAG(TTP)/(DD1**3)
  CURVA=1/((1-UU)*(1/CUR(I_P,ILL,I))+
 +     UU*(1/CUR(I_P,ILL,I+1)))
  !IF ((JOIN(I_P)==1).AND.(NR(I_P)/=1).AND.(UU/=0)) THEN;DIS2=DIS*NR(I_P)*CURVA*0.01
  !ELSE;DIS2=0;END IF
```

```
   DIS2=0
   IF (NR(I_P)==1.0D0) THEN
     IF((CUSPTEM(I_P,ILL,NTEM(I_P,ILL))==-2).OR.
  +  (CUSPTEM(I_P,ILL,NTEM(I_P,ILL))==2)) THEN
       NN=1.7D0*G*CUSPTEM(I_P,ILL,NTEM(I_P,ILL))*RT/2.0D0/DD1
     END IF
   ELSE
     IF((CUSP(I_P,ILL,I)==-2).OR.(CUSP(I_P,ILL,I)==2)) THEN
       NN=1.6D0*G*CUSP(I_P,ILL,I)*RT/2.0D0/DD1
     END IF
   END IF
   !CO=F(INT(DACOS(RT*B/DD1/MAG(B))/PI*180))/2.0D0
   NN=D%V(3)**UV(RT)
   QL=PTEM(I_P,ILL,NTEM(I_P,ILL))
   Q=TRANS(ES(I_P))*QL+ORIGIN(I_P)
!   WRITE(*,*) "ILL  Q   ", ILL,Q
 ! WRITE(*,*)"RT,NN,DELTA_P  ",RT,NN,DELTA_P
   CALL DISPLACE_NODES(Q,UV(RT),NN,DELTA_P)
   !Calculate the displacement
   IF (CURVA<0.5/RR0(I_P,ILL)) THEN
     CALPHA=UV(RT)*UV(BURGERS(I_P,ILL))
   ELSE
     CALPHA=UV(PL(I_P,ILL,N_NODE(I_P,ILL))-PL(I_P,ILL,1))*
  +        UV(BURGERS(I_P,ILL))
   END IF
   CALPHA=(1-CURVA*RR0(I_P,ILL))*UV(RT)*UV(BURGERS(I_P,ILL))
  +      +CURVA*RR0(I_P,ILL)*UV(PL(I_P,ILL,N_NODE(I_P,ILL))-
  +       PL(I_P,ILL,1))*UV(BURGERS(I_P,ILL))
!   WRITE(*,*)"ILL I RT   ",
 ! +          ILL, I, RT
 !  WRITE(*,*)"ILL CALPHA  CURVA  ", ILL,CALPHA, CURVA
 !  WRITE(*,*)"SELF_DELTA(CALPHA,CURVA)  ",SELF_DELTA(CALPHA,CURVA)
   INCREMENT=DELTA_P!+SELF_DELTA(CALPHA,CURVA)
    IF (INCREMENT>0) THEN
     INCREMENT=0
    END IF
!   WRITE(*,*)"ILL I DELTA_P INCREMENT  NN",
 ! +         ILL,I,DELTA_P, INCREMENT, NN
   PTEM(I_P,ILL,NTEM(I_P,ILL))=PTEM(I_P,ILL,NTEM(I_P,ILL))+
  +                        NR(I_P)*INCREMENT*NN

   !+  MS*NR(I_P)*(CO*CURVA-SIGMA/MU-DIS2)*NN


   END SUBROUTINE DISPLACEMENT

   !************************************************************!

       SUBROUTINE DOUBLE_INTEGRAL(I_P1,I_P2,IL1,IL2,
  +                                          ID_NODE_L1,ID_NODE_L2, F4)

   USE VECTORS
   USE VARIABLES

   TYPE(VECTOR)::P1,P2,T_G1,T_G2,P_L1,P_L2,T_L1,T_L2,TT1,TT2,B1,B2,E
   INTEGER,INTENT(IN)::I_P1,I_P2,IL1,IL2,ID_NODE_L1,ID_NODE_L2
   double precision, dimension(300)::pos,wt
   DOUBLE PRECISION, INTENT(OUT)::F4
   DOUBLE PRECISION::R,TM1,TM2, FACT1, U1,U2,
  +      WEIGHT,B1T1,B2T2,B1B2,T1T2,B1T2,B2T1,B1E,B2E


   CALL quadrature(MAX_QUAD,pos,wt)
   F4=0.D0

     B1    =BURGERS(I_P1,IL1)
     B2    =BURGERS(I_P2,IL2)
            DO I=1,MAX_QUAD
          U1=(POS(I)+1)/2.0D0
       CALL GET_POINT_TAN(U1,P_L1,T_L1,I_P1,IL1,ID_NODE_L1)
       P1  =TRANS(ES(I_P1))*P_L1+ORIGIN(I_P1)
       T_G1=TRANS(ES(I_P1))*T_L1
       TM1              =MAG(T_G1)
       TT1              =UV(T_G1)

      DO J=1,MAX_QUAD
       U2=(POS(J)+1)/2.0D0
       WEIGHT = WT(I)*WT(J)
       CALL GET_POINT_TAN(U2,P_L2,T_L2,I_P2,IL2,ID_NODE_L2)

       P2=TRANS(ES(I_P2))*P_L2+ORIGIN(I_P2)
       T_G2=TRANS(ES(I_P2))*T_L2
       TM2              =MAG(T_G2)
       TT2              =UV(T_G2)

     R=MAG(P1-P2)
     IF (R<RMIN) THEN
       R=RMIN
     END IF

     E      =(P1-P2)/R

     FACT1            =  G*TM1*TM2/8./PI/R
            B1T1      =B1*TT1
            B2T2      =B2*TT2
            B1B2      =B1*B2
            B1T2      =B1*TT2
            B2T1      =B2*TT1
            T1T2      =TT1*TT2
            B1E       =B1*E
            B2E       =B2*E

            F4        =F4+FACT1*(2.*B1T1*B2T2-FACT2*B1B2*T1T2+FACT3*B1T2*B2T1-
  +                                        FACT2*T1T2*B1E*B2E)*WEIGHT*INTERVAL2
               END DO
     END DO
   END SUBROUTINE DOUBLE_INTEGRAL

   !*************************************************************!

       SUBROUTINE EQUILIBRIUM
   USE VARIABLES
   USE VECTORS
   DOUBLE PRECISION::CT,ST,DT,LL,PLL,RATIO,PDANG,NDANG,DANG1,
  +               MIN_DIST,DIST
   DOUBLE PRECISION,DIMENSION(2)::NDANG1
   TYPE(VECTOR)::DANG,T0
   TYPE(VECTOR),DIMENSION(2)::NORV,NORV1 !Changed to 2D array
   INTEGER::I,J,J1,I_P1,IL1,I_P2,IL2,I_P3,IL3


   !FIND DIPOLES
   DO I_P1=1,N_PLANE
     DO IL1=1,NLOOP(I_P1)
      MIN_DIST=1000
      DO I_P2=1,N_PLANE
       DO IL2=1,NLOOP(I_P2)
            DIST=MAG(PL(I_P1,IL1,2)-PL(I_P2,IL2,2))
            IF ((DIST>0).AND.(DIST<MIN_DIST)) THEN
              MIN_DIST=DIST
              PROP_DIPOLE(I_P1,IL1,2)=I_P2
              PROP_DIPOLE(I_P1,IL1,3)=IL2
          END IF
       END DO
       END DO
       IF (MIN_DIST>NEAREST_DIST) THEN
       PROP_DIPOLE(I_P1,IL1,2)=0
       PROP_DIPOLE(I_P1,IL1,3)=0
       END IF
     END DO
  END DO
  N_DIPOLE=0
  DO I_P1=1,N_PLANE
     DO IL1=1,NLOOP(I_P1)
       IF ((PROP_DIPOLE(I_P1,IL1,2)>0).AND.
  +       (PROP_DIPOLE(I_P1,IL1,1)==0)) THEN
       I_P2=PROP_DIPOLE(I_P1,IL1,2)
       IL2=PROP_DIPOLE(I_P1,IL1,3)
       IF ((PROP_DIPOLE(I_P2,IL2,2)==I_P1).AND.
  +         (PROP_DIPOLE(I_P2,IL2,3)==IL1)) THEN
          N_DIPOLE=N_DIPOLE+1
          PROP_DIPOLE(I_P1,IL1,1)=N_DIPOLE
          PROP_DIPOLE(I_P2,IL2,1)=N_DIPOLE
       ELSE
          DO
            I_P3=PROP_DIPOLE(I_P2,IL2,2)
            IL3=PROP_DIPOLE(I_P2,IL2,3)
            IF ((PROP_DIPOLE(I_P3,IL3,2)==I_P2).AND.
  +         (PROP_DIPOLE(I_P3,IL3,3)==IL2)) EXIT
            I_P2=PROP_DIPOLE(I_P3,IL3,2)
            IL2=PROP_DIPOLE(I_P3,IL3,3)
          END DO
          N_DIPOLE=N_DIPOLE+1
          PROP_DIPOLE(I_P2,IL2,1)=N_DIPOLE
          PROP_DIPOLE(I_P3,IL3,1)=N_DIPOLE
       END IF
       END IF
     END DO
  END DO

  !FIND THE INITIAL GEOMETRY OF EACH LOOP (EXCLUDING THE INTERACTION BETWEEN DIPOLE)
  DO I_P1=1,N_PLANE
     DO IL1=1,NLOOP(I_P1)
       CALL SINGLE_EQUILIBRIUM(IP1,IL1)
     END DO
  END DO

  !START DIPOLE DO LOOP

  DT=PI/360
  IL=1
  DO I_P=1,2
     NQ(I_P)=180
     DO J=1,NQ(I_P)
       QD(I_P,J)=PL(I_P,IL,1)+(J/180D0)*(PL(I_P,IL,3)-PL(I_P,IL,1))
       QT(I_P,J)=PL(I_P,IL,3)-PL(I_P,IL,1)
     END DO
  END DO
  DO I_P=1,2
     NORV(I_P)=G*UV(D%V(3)**(PL(I_P,IL,2)-PL(I_P,IL,1)))
  END DO
  DO J1=1,3
     DO I_P=1,2
       RATIO=1D0
       PDANG=0.0D0
       DO
       NORV1(I_P)=NORV(I_P)
       I=1
```

```
        LL=0
        DO
          PLL=LL
          QG=TRANS(ES(I_P))*QD(I_P,I)+ORIGIN(I_P)
          CALL CURVATURE(D%V(3)**NORV(I_P),G*NORV(I_P))
          QD(I_P,I+1)=QD(I_P,I)-R0*NORV(I_P)+DT**(R0*NORV(I_P))
          QT(I_P,I+1)=D%V(3)**(DT**(R0*NORV(I_P)))
          NORV(I_P)=DT**NORV(I_P)
          I=I+1
          LL=MAG(QD(I_P,I)-QD(I_P,1))
          IF ((LL>MAG(PL(I_P,IL,3)-QD(I_P,1))).OR.(LL<PLL)) EXIT
          !INSTABILITY CONDITION LL<PLL
        END DO
        DANG=UV(QD(I_P,I)-QD(I_P,1))**UV(PL(I_P,IL,3)-QD(I_P,1))
        DANG1=UV(QD(I_P,I)-QD(I_P,1))*UV(PL(I_P,IL,3)-QD(I_P,1))
        NDANG=DACOS(DANG1)*DANG%V(3)/ABS(DANG%V(3))
        IF (PDANG==0.0D0) THEN
          RATIO=1
        ELSE
          RATIO=((NDANG-PDANG)/(PDANG/RATIO))
        END IF
        NORV(I_P)=(NDANG/RATIO)**NORV1(I_P)
        PDANG=NDANG
        !WRITE(*,*)"DANG1  DANG%V(3)",DANG1,DANG%V(3)
        IF ((MAG(DANG)<0.05D0).AND.(DANG1>0)) EXIT
      END DO
      NDANG1(I_P)=NDANG
      NQ(I_P)=I
      WRITE(*,*)"I_P  ",I_P
    END DO
  END DO

  !WRITE(*,*)"LINE 50"
  !IF (LL<MAG(QD(I_P,NQ(I_P))-QD(I_P,1))) THEN
  !  WRITE(*,*)"(2)LOOP IS UNSTABLE."
  !END IF


  WRITE(6,FMT=16) "3DPOLY"
16 FORMAT(A6)

  DO I_P=1,2
    DO J=1,NQ(I_P)
      QG=TRANS(ES(I_P))*QD(I_P,J)+ORIGIN(I_P)
      IF ((I_P/=1).AND.(J==1)) THEN
        WRITE(6,FMT=18) "  ",QG%V(1),",",QG%V(2),",",QG%V(3)
18        FORMAT(A2,E12.6,A1,E12.6,A1,E12.6)
      ELSE
        WRITE(6,FMT=17) QG%V(1),",",QG%V(2),",",QG%V(3)
17        FORMAT(E12.6,A1,E12.6,A1,E12.6)
      END IF
    END DO
  END DO

  END SUBROUTINE EQUILIBRIUM

  !*************************************************************!

        SUBROUTINE GET_POINT_TAN(UU,P1,TT1,IP_LOC,IL_LOC,ID_NODE)
  USE VECTORS; USE VARIABLES
  DOUBLE PRECISION,INTENT(IN)::UU
  INTEGER,INTENT(IN)::IP_LOC,IL_LOC,ID_NODE
  DOUBLE PRECISION::F1,F2,F3,F4,F1P,F2P,F3P,F4P
  TYPE(VECTOR),INTENT(OUT)::P1,TT1

  IF (LOOPTYPE(I_P,IL)==1) THEN
  SEG_TYPE="QUINTIC"
 ELSE
  SEG_TYPE="CUBIC"
 END IF

 IF (INTERACTION_TYPE=="DIPOLE") THEN
  SEG_TYPE="COMP_ARC"
 ELSE IF (INTERACTION_TYPE=="SINGLE_EQUILIBRIUM") THEN
  SEG_TYPE="LINEAR"
 END IF

 SELECT CASE(SEG_TYPE)
 CASE("LINEAR") !Linear Segment
  P1=(1-UU)*PL(IP_LOC,IL_LOC,ID_NODE)+
+     UU*PL(IP_LOC,IL_LOC,ID_NODE+1)
  TT1=PL(IP_LOC,IL_LOC,ID_NODE+1)-PL(IP_LOC,IL_LOC,ID_NODE)
 CASE("ARC") !Circular Arc

 CASE("CUBIC") !Cubic Spline
  F1 =2*UU**3-3*UU*UU+1
  F2 =-2*UU**3+3*UU*UU
  F3 = UU**3-2*UU*UU+UU
  F4 = UU**3-UU*UU

  F1P= 6*UU**2-6*UU
  F2P=-6*UU**2+6*UU
  F3P=3*UU**2-4*UU+1
  F4P=3*UU**2-2*UU
  CALL TANGENT_VEC
  P1=F1*PL(IP_LOC,IL_LOC,ID_NODE)+F2*PL(IP_LOC,IL_LOC,ID_NODE+1)
+   +F3*TL(IP_LOC,IL_LOC,ID_NODE)+F4*TL(IP_LOC,IL_LOC,ID_NODE+1)

  TT1=F1P*PL(IP_LOC,IL_LOC,ID_NODE)+
+   F2P*PL(IP_LOC,IL_LOC,ID_NODE+1)
+   +F3P*TL(IP_LOC,IL_LOC,ID_NODE)+F4P*TL(IP_LOC,IL_LOC,ID_NODE+1)


 CASE("QUINTIC") !Quintic Spline
  CALL PLOT(IL,ID_NODE_L,UU)
  P1=R_PLOT
  CALL TANG(IL,ID_NODE_L,UU)
  TT1=RT
  !WRITE(*,*)"P1 TT1   ",P1,TT1

 CASE("COMP_ARC") !Composite circular arc
  P1=QD(IP_LOC,INT(NQ(IP_LOC)*UU))
  TT1=QT(IP_LOC,INT(NQ(IP_LOC)*UU))
 END SELECT
 END SUBROUTINE GET_POINT_TAN

!*************************************************************!

     SUBROUTINE INITIAL

  USE VARIABLES
  USE VECTORS



 CUR1  =0.0005
 CU1    =0.0005
 DENSITY           =1.4
 NR1    =5
 H                    =0.5
 G                    =-1.0
 AS      =5.0
```

```
FACT2           = 2./(1.-NU)
FACT3           = 4.*NU/(1.-NU)

 !prepare identity & Nzero matrices , then and Nzero vector

  do i=1,3
    do j=1,3
      D%v(i)%v(j)=int(i/j)*int(j/i)
      ZERO%V(I)%V(J)=0.0D0
    end do
  end do

IC=1
VALSPLIT=0
III=0

! find coordinate transformation matrices for all planes

DO I_P=1,N_PLANE
  ES(I_P)%v(3)=UV(MILLER(I_P))
  ES(I_P)%v(1)=UV(d%v(3)**MILLER(I_P))
  ES(I_P)%v(2)=ES(I_P)%v(3)**ES(I_P)%v(1)
END DO

! determine initial displacements on all linear segments, this
! can be determined from applied stress

DO I_P=1,N_PLANE
  NR(I_P)=1
  VALSTOP1(I_P)=1
  DO IL=1,NLOOP(I_P)
    IF (LOOPTYPE(I_P,IL)==1) THEN
      N_NODE(I_P,IL)=3
      PL(I_P,IL,3)=PL(I_P,IL,2)
      PL(I_P,IL,2)=H*(PL(I_P,IL,3)+PL(I_P,IL,1))-
+               0.1D0*D%V(3)**(PL(I_P,IL,3)-PL(I_P,IL,1))
      CALL CURVATURE(UV(PL(I_P,IL,3)-PL(I_P,IL,1)),D%V(3)**
+                UV(PL(I_P,IL,3)-PL(I_P,IL,1)))
      RR0(I_P,IL)=R0
      WRITE(*,*) "RR0(I_P,IL)  I_P",RR0(I_P,IL) ,I_P
    ELSE
    END IF
  END DO
  DO I=1,MAX_LOOP
    DO J=1,MAX_NODE
      CUSP(I_P,I,J)=0
      CUSPTEM(I_P,I,J)=0
    END DO
  END DO
  DO I=1,NLOOP(I_P)
    VALID(I_P,I)=1
    DO J=1,3
      PROP_DIPOLE(I_P,I,J)=0
    END DO
  END DO
  DO I=1,NOBS(I_P)
    VALOBS(I_P,I)=1
  END DO
  DO IL=1,NLOOP1(I_P)
      VALID1(I_P,IL,1)=1
      NSEG(I_P,IL)=1
      VALSTOP(I_P,IL)=1
  END DO
END DO

DIS=MS*SIG_APP%V(1)%V(1)
```

```
      DO I_P=1,N_PLANE
       JOIN(I_P)=0
       DO I=1,MAX_LOOP
             DO J=1,MAX_LOOP
               PMD(I_P,I,J)=3*DIS
             END DO
           END DO
      END DO

      END SUBROUTINE INITIAL

      !*************************************************************!

      SUBROUTINE interaction_geom_writer(FORCE_KIND) !!RUDY 3_9

      use vectors

            CHARACTER(LEN=10), INTENT(IN) :: FORCE_KIND  !!RUDY 3_9


      CHARACTER (len=8):: CIRCLE, POLYGON, FRS
      CHARACTER ::        DEFECT_KIND*20, ENDDEFECT*9
      double precision::pii
      INTEGER :: number_of_circles, number_of_polygons, number_of_frs,
     +          PLANE_NUMBER, NODES , i, k, defect ,CIRCLE1
            double precision :: pheta, circ_radius,KAPPA


       Type(vector):: BURGERL, plane_miller,
     +              ORIGINL, circle_cntr, P_Local,T_LOCAL


           CIRCLE1=0
             pii = 2.d0*dasin(1.d0)
           number_of_circles  = 0
           number_of_polygons = 0
           number_of_frs      = 0


      DO WHILE (DEFECT_KIND .NE. 'DEFECT' )
            READ (3,*) DEFECT_KIND
              IF (DEFECT_KIND .EQ. 'DEFECT') THEN
                     DO WHILE (DEFECT_KIND .NE. 'ENDDEFECT')



            READ (3,*) DEFECT_KIND
            IF (DEFECT_KIND .EQ. 'POLYGON') GOTO 1
            IF (DEFECT_KIND .EQ. 'FRS')     GOTO 2
            IF (DEFECT_KIND .EQ. 'ENDDEFECT') GOTO 777


   !------ READING CIRCLE INPUT


      IF(DEFECT_KIND.EQ.'circle') THEN

                     NUMBER_OF_CIRCLES = NUMBER_OF_CIRCLES + 1
                     defect = 0
                     READ (3,*) DEFECT_KIND, PLANE_NUMBER, NODES,
     +               plane_miller, BURGERL, ORIGINL, circ_radius,
     +               circle_cntr%v(1), circle_cntr%v(2)
       radi=circ_radius
            WRITE(2,FMT=101)PLANE_NUMBER
            WRITE(2,FMT=102)NUMBER_OF_CIRCLES,plane_miller, ORIGINL,CIRCLE1
            WRITE(2,FMT=103)BURGERL, NODES, DEFECT
            IF (FORCE_KIND == 'LINE_FORCE') THEN
            WRITE(4,FMT=106)NODES,ORIGINL,plane_miller,BURGERL,PLANE_NUMBER,
   +                              number_of_circles    !!RUDY 3_1
     END IF

      DO k = 0, NODES-1
           pheta = (2.* pii/NODES)*k
       P_local%v(1) = circle_cntr%v(1) + circ_radius * dcos(pheta)
       P_local%v(2) = circle_cntr%v(2) + circ_radius * dsin(pheta)
       P_local%v(3) = 0.0

       T_LOCAL%V(1)=-dsin(pheta)                              !!RUDY 3_1
       T_LOCAL%V(2)= dcos(pheta)                              !!RUDY 3_1
       T_LOCAL%V(3)= 0                                                !!RUDY 3_1
       KAPPA=1/circ_radius                                    !!RUDY 3_1

             WRITE(2,FMT=104)P_local

             !write(2,FMT=104)T_LOCAL                         !!RUDY 5/6



             IF (FORCE_KIND == 'LINE_FORCE') THEN
              WRITE(4,FMT=105)P_local,T_LOCAL,KAPPA           !!RUDY 3_1
             END IF

      END DO


    END IF



              REWIND(4)

!-----READING POLYGON INPUT ------------------------------------

  1        IF(DEFECT_KIND .EQ. 'polygon' ) THEN
                        defect = 0
                        READ (3,*) DEFECT_KIND, PLANE_NUMBER, NODES,
   +                plane_miller, BURGERL, ORIGINL
            WRITE(2,FMT=101)PLANE_NUMBER
            WRITE(2,FMT=102)number_of_polygons, plane_miller,ORIGINL,CIRCLE1
            WRITE(2,FMT=103)BURGERL, NODES, DEFECT
     DO k = 1, NODES
                        READ (3,*) J, P_local%v(1),P_local%v(2)
                  WRITE(2,FMT=104)P_local%v(1),P_local%v(2)
            END DO
          END IF


!-----READING FRS INPUT ------------------------------------

  2        IF(DEFECT_KIND .EQ. 'frs' ) THEN
                        number_of_frs = number_of_frs + 1
                        defect = 1
                        READ (3,*) DEFECT_KIND, PLANE_NUMBER, NODES,
   +                plane_miller, BURGERL, ORIGINL, circ_radius,
   +                circle_cntr%v(1), circle_cntr%v(2)
            WRITE(2,FMT=101)PLANE_NUMBER
            WRITE(2,FMT=102)number_of_frs, plane_miller, ORIGINL,CIRCLE1
            WRITE(2,FMT=103)BURGERL, NODES, DEFECT
     DO k = 1, 2
                         READ (3,*) J, P_local%v(1),P_local%v(2)
                   WRITE(2,FMT=104)P_local%v(1),P_local%v(2)
               END DO
            END IF

!------------------------------------------------------------

        END DO
      END IF
    END DO


  777 CONTINUE




 101  FORMAT(i3)
 102  FORMAT(i3, ","  ,3(es10.3,","), 3(es10.3,","),I3)
 103  FORMAT(3(es10.3,","), 2(i3,","))
 104  FORMAT(3(es10.3,","))
 105  FORMAT(7(es10.3,","))     !!RUDY 3_1
 106  FORMAT(I3,2X,9(es10.3,2X),I4,2X,I4)!!RUDY 3_1
      WRITE(*,*)"HELLO"
    END SUBROUTINE interaction_geom_writer


              !**************************************************************!

                   SUBROUTINE JUNCTION(IL1,IL2)
    USE VECTORS;USE VARIABLES
    INTEGER,INTENT(IN)::IL1,IL2
    INTEGER::Q1,Q2,J1,J2,J3,J4,I,I1,I2,I3,Q3,Q4
    DOUBLE PRECISION::TEM,DOT1,DOT2,DOT3,DOT4,DIST1,DIST2,DIST3,DIST4
    TYPE(VECTOR)::X,TA,TA1
    JOIN(I_P)=1;Q1=KN(1);Q2=KN(3);X=H*(PL(I_P,IL1,Q1)+PL(I_P,IL2,Q2))
    TA=H*(TL(I_P,IL1,Q1)-TL(I_P,IL2,Q2));TA1=D%V(3)**TA
    J1=1;J2=1;J3=1;J4=1
    DO
      IF (Q1-J1<1) THEN;J1=J1-N_NODE(I_P,IL1);END IF
      DIST1=TA*TL(I_P,IL1,Q1-J1)
      IF (DIST1>0) THEN;J1=J1+1;DOT1=DIST1;END IF
      IF (Q2+J2>N_NODE(I_P,IL2)) THEN;J2=J2-N_NODE(I_P,IL2);END IF
      DIST2=G*TA*TL(I_P,IL2,Q2+J2)
      IF (DIST2>0) THEN;J2=J2+1;DOT2=DIST2;END IF
      IF (Q2-J3<1) THEN;J3=J3-N_NODE(I_P,IL2);END IF
      DIST3=G*TA*TL(I_P,IL2,Q2-J3)
      IF (DIST3>0) THEN;J3=J3+1;DOT3=DIST3;END IF
      IF (Q1+J4>N_NODE(I_P,IL1)) THEN;J4=J4-N_NODE(I_P,IL1);END IF
      DIST4=TA*TL(I_P,IL1,Q1+J4)
      IF (DIST4>0) THEN;J4=J4+1;DOT4=DIST4;END IF
      IF ((DIST1<0).AND.(DIST2<0).AND.(DIST3<0).AND.(DIST4<0)) EXIT
    END DO
    I1=0;I1=I1+1
    IF (DIST1<-1.0D-2) THEN
      CALL PLOT(IL1,Q1-J1,-DIST1/(DOT1-DIST1))
      PTEM(I_P,IL1,I1)=R_PLOT
    ELSE
      PTEM(I_P,IL1,I1)=PL(I_P,IL1,Q1-J1);J1=J1+1
    END IF
    TTEM(I_P,IL1,I1)=G*TA1;I1=I1+1
    PTEM(I_P,IL1,I1)=X-CU1*DIS*TA;TTEM(I_P,IL1,I1)=TA
    I1=I1+1;PTEM(I_P,IL1,I1)=X+CU1*DIS*TA
    TTEM(I_P,IL1,I1)=TA;I1=I1+1
    IF (DIST4<-1.0D-2) THEN
      CALL PLOT(IL1,Q1+J4-1,DOT4/(DOT4-DIST4))
      PTEM(I_P,IL1,I1)=R_PLOT
    ELSE
      PTEM(I_P,IL1,I1)=PL(I_P,IL1,Q1+J4);J4=J4+1
    END IF
    TTEM(I_P,IL1,I1)=TA1
    IF(Q1-J1>Q1+J4) THEN;Q3=Q1-J1;ELSE;Q3=Q1-J1+N_NODE(I_P,IL1);END IF
    DO I=Q1+J4,Q3
      I1=I1+1
      IF (I>N_NODE(I_P,IL1)) THEN;Q4=N_NODE(I_P,IL1);ELSE;Q4=0;END IF
      PTEM(I_P,IL1,I1)=PL(I_P,IL1,I-Q4)
      CUSPTEM(I_P,IL1,I1)=CUSP(I_P,IL1,I-Q4)
      TTEM(I_P,IL1,I1)=TL(I_P,IL1,I-Q4)
    END DO
    NTEM(I_P,IL1)=I1
    I1=0;I1=I1+1
    IF (DIST3<-1.0D-2) THEN
      CALL PLOT(IL2,Q2-J3,-DIST3/(DOT3-DIST3))
      PTEM(I_P,IL2,I1)=R_PLOT
    ELSE
      PTEM(I_P,IL2,I1)=PL(I_P,IL2,Q2-J3);J3=J3+1
    END IF
    TTEM(I_P,IL2,I1)=TA1;I1=I1+1
    PTEM(I_P,IL2,I1)=X+CU1*DIS*TA;TTEM(I_P,IL2,I1)=G*TA
    I1=I1+1;PTEM(I_P,IL2,I1)=X-CU1*DIS*TA
    TTEM(I_P,IL2,I1)=G*TA;I1=I1+1
    IF (DIST2<-1.0D-2) THEN
      CALL PLOT(IL2,Q2+J2-1,DOT2/(DOT2-DIST2))
      PTEM(I_P,IL2,I1)=R_PLOT
    ELSE
      PTEM(I_P,IL2,I1)=PL(I_P,IL2,Q2+J2);J2=J2+1
    END IF
    TTEM(I_P,IL2,I1)=G*TA1
    IF(Q2-J3>Q2+J2) THEN;Q3=Q2-J3;ELSE;Q3=Q2-J3+N_NODE(I_P,IL2);END IF
    DO I=Q2+J2,Q3
      I1=I1+1
      IF (I>N_NODE(I_P,IL2)) THEN;Q4=N_NODE(I_P,IL2);ELSE;Q4=0;END IF
      PTEM(I_P,IL2,I1)=PL(I_P,IL2,I-Q4)
      CUSPTEM(I_P,IL2,I1)=CUSP(I_P,IL2,I-Q4)
      TTEM(I_P,IL2,I1)=TL(I_P,IL2,I-Q4)
    END DO
    NTEM(I_P,IL2)=I1
    PL=PTEM;TL=TTEM;N_NODE=NTEM;CUSP=CUSPTEM
    CUSP(I_P,IL1,2)=-2;CUSP(I_P,IL1,3)=2
    CUSP(I_P,IL2,2)=-2;CUSP(I_P,IL2,3)=2
    CUSP(I_P,IL1,1)=0;CUSP(I_P,IL1,4)=0
    CUSP(I_P,IL2,1)=0;CUSP(I_P,IL2,4)=0
    CALL TAN_NOR(IL1);CALL TAN_NOR(IL2)
    END SUBROUTINE JUNCTION

    !**************************************************************!

       SUBROUTINE LINE_INTEGRAL(Q,F1,F2,F3,IP_LOC,IL_LOC,ID_NODE)

    USE VECTORS
    USE VARIABLES
    TYPE(VECTOR), INTENT(IN)::Q
    double precision, dimension(300)::pos,wt
    DOUBLE PRECISION::R,S,TM, FACTOR1, FACTOR2, FACTOR3
    INTEGER,INTENT(IN)::IP_LOC,IL_LOC,ID_NODE
    TYPE(VECTOR)::E,P,P_L,T,T_L,T_G,S1,S2,S3
    TYPE(MATRIX)::F,R2,TBE,BET,ETB
    TYPE(MATRIX),INTENT(OUT)::F1,F2
    TYPE(VECTOR),INTENT(OUT)::F3
    TYPE(TENSOR)::R3
```

```
CALL quadrature(MAX_QUAD,pos,wt)
F1=ZERO
F2=ZERO
F3=ZERO%V(1)
DO I=1,MAX_QUAD
  U=(POS(I)+1)/2.0D0
  CALL GET_POINT_TAN(U,P_L,T_L,IP_LOC,IL_LOC,ID_NODE)
  P=TRANS(ES(IP_LOC))*P_L+ORIGIN(IP_LOC)
  T_G=TRANS(ES(IP_LOC))*T_L
  TM=MAG(T_G)
  T=UV(T_G)
  R=MAG(Q-P)
  IF (R<RMIN) THEN
    R=RMIN
  END IF

  E=UV(Q-P)
  TBE=T**BURGERS(IP_LOC,IL_LOC)/E
  TBE=TBE+TRANS(TBE)
  BET=BURGERS(IP_LOC,IL_LOC)**E/T
  BET=BET+TRANS(BET)
  ETB=E**T/BURGERS(IP_LOC,IL_LOC)
  ETB=ETB+TRANS(ETB)
  S=T**BURGERS(IP_LOC,IL_LOC)*E

  FACTOR1=INTERVAL*WT(I)*TM/(4*PI*R*R)
  FACTOR2=FACTOR1/2.0D0
  FACTOR3=FACTOR1*R

  F1=F1+FACTOR1*(BET+(TBE-S*D-3*S*(E/E))/(1-NU))
  F2=F2+FACTOR2*((NU*TBE-3*S*(E/E)+S*D-ETB)/(1-NU))
  F3=F3+FACTOR3*((ARB**E*T/(1+ARB*E))*BURGERS(IP_LOC,IL_LOC)+
+     ((1-2*NU)*T**BURGERS(IP_LOC,IL_LOC)+S*E)/2.0D0/(1-NU))

END DO

!  WRITE(*,*) "IP_LOC R Q P",IP_LOC,R,Q,P
!WRITE(*,*)"IP_LOC,P,Q, F1  ",IP_LOC,P,Q, F1
END SUBROUTINE LINE_INTEGRAL

!*************************************************************!

      SUBROUTINE LOOP
USE VARIABLES; USE VECTORS
IMPLICIT NONE
TYPE(VECTOR)::A1
INTEGER::I,J,K
DOUBLE PRECISION,EXTERNAL::MIN_DIST

DO I_P=1,N_PLANE

  !Set initial values

  MIN_NR=3*DIS


           !Plot the current loops
  DO IL=1,NLOOP(I_P)
    IF (VALID(I_P,IL)==1) THEN
      CALL TAN_NOR(IL)
      DO I=1,N_NODE(I_P,IL)-LOOPTYPE(I_P,IL)
        DO K=0,NPLOT
          U=1.0*K/NPLOT
          CALL PLOT(IL,I,U)
          !CALL TRANS1(R,GV,MILLER(I_P),ORIGIN(I_P),1)
          GV=TRANS(ES(I_P))*R_PLOT+ORIGIN(I_P)
          IF (((I_TIME/=1).OR.(IL/=1).OR.(I_P/=1)).AND.(I==1).
 +            AND.(K==0))THEN
           WRITE(6,FMT=6) "  ",GV%V(1),",",GV%V(2),",",GV%V(3)
6            FORMAT(A2,E12.6,A1,E12.6,A1,E12.6)
          ELSE
           WRITE(6,FMT=7) GV%V(1),",",GV%V(2),",",GV%V(3)
7            FORMAT(E12.6,A1,E12.6,A1,E12.6)
          END IF
         END DO
        END DO
        DO I=1,N_NODE(I_P,IL)
         !CALL TRANS(PL(I_P,IL,I),MI1,MI2,MI3,ORIGIN(I_P))
         WRITE(5,FMT=8)100*PL(I_P,IL,I)%V(1),",",
 +            100*PL(I_P,IL,I)%V(2)
8        FORMAT(E12.6,A1,E12.6)
        END DO
       END IF
       IF (I_TIME==N_TIMES) THEN
        WRITE(*,*)"IL,I",IL,I
       END IF
      END DO

      MIN_NR=NR(I_P)
      NLOOP_TEM=NLOOP(I_P)
      DO I=1,MAX_LOOP
       DO J=1,MAX_LOOP
        L_L(I_P,I,J)=0
       END DO
      END DO
      DO I=1,NLOOP_TEM
       DO J=I,NLOOP_TEM
        IF ((VALID(I_P,I)==1).AND.(VALID(I_P,J)==1)) THEN
         MD=MIN_DIST(I,J)
         IF ((MD<2*DIS).AND.(MD>1.0D-4)) THEN
          IF (MD>2*DIS/NR1) THEN
           IF (MD/DIS/4<MIN_NR) THEN
            MIN_NR=MD/DIS/4
           END IF
           IF (PMD(I_P,I,J)>2*DIS) THEN
            L_L(I_P,I,J)=1
           END IF
          ELSE
           IF (PMD(I_P,I,J)>2*DIS) THEN
            L_L(I_P,I,J)=1
            IF (MD/DIS/4<MIN_NR) THEN
             MIN_NR=MD/DIS/4
            END IF
           ELSE
            IF ((LOOPTYPE(I_P,I)==0).AND.(LOOPTYPE(I_P,J)==0))
 +             THEN
             L_L(I_P,I,J)=3
            ELSE
             WRITE(*,*)"###############################"
             L_L(I_P,I,J)=2
            END IF
            MIN_NR=NR(I_P)/DENSITY
           END IF
          END IF
         END IF
         PMD(I_P,I,J)=MD
        END IF
       END DO
      END DO

      DO I=1,NLOOP_TEM
       DO J=I,NLOOP_TEM
        IF (L_L(I_P,I,J)==1) THEN
         MD=MIN_DIST(I,J)
         CALL REFINE(I,J)
        END IF
        IF ((L_L(I_P,I,J)==2).OR.(L_L(I_P,I,J)==3)) THEN
         DIS1=MIN_DIST(I,J)
         IF (L_L(I_P,I,J)==2) THEN
          CALL ANNIHILATE(I,J)
         ELSE
          CALL JUNCTION(I,J)
         END IF
         WRITE(6,FMT=9)" COLOR ",IC," 3DPOLY 0,0,0";IC=IC+2
 9        FORMAT(A7,I1,A13)
        END IF
       END DO
      END DO
      NR(I_P)=MIN_NR
      DO IL=1,NLOOP(I_P)
       IF (VALID(I_P,IL)==1) THEN
        CALL ARRANGE(IL)
       END IF
      END DO
      IF (JOIN(I_P)==1) THEN
       NR(I_P)=DENSITY*NR(I_P)
      END IF
      IF (NR(I_P)==1.0D0) THEN
       CUSP=CUSPTEM
      END IF
      IF (NR(I_P)>1.0D0) THEN
       NR(I_P)=1;JOIN(I_P)=0
      END IF
     END DO
     PL=PTEM;N_NODE=NTEM;TL=TTEM;NL=NT
     END SUBROUTINE LOOP

     !**************************************************************!

        SUBROUTINE LOOP_OBS
     USE VARIABLES; USE VECTORS
     TYPE(VECTOR)::A1,P1,P2,P3,P4,SP,EP,V1,LV
     INTEGER::I,I1,I2,J,JL,IS,NN,IS1
     DOUBLE PRECISION::ANG1,EM,DD,D1,CA,SA

     ! .AND.(VALSTOP1(I_P)==1))
     DO I_P=1,N_PLANE
      !DETERMINE THE MAXIMUM RADIUS OF CURVATURE (>R0)
      IF (VALSTOP1(I_P)==1) THEN

       VALSTOP1(I_P)=0
       DO IL=1,NLOOP1(I_P)
        IF (VALSTOP(I_P,IL)==1) THEN

         IS=1;VALSPLIT=0
         DO
          ILSPLIT=0
          CALL CEN_CUR(IL,IS);CALL MAX_RAD(IL,IS)
          RR1(I_P,IL,IS)=RMAX1
          IF (ILSPLIT/=0) THEN
           J=0
           DO
            J=J+1
            IF ((VALID1(I_P,IL,J)==0).OR.(J-1==NSEG(I_P,IL)))
 +            EXIT
           END DO
           IF (J-1==NSEG(I_P,IL)) THEN
            NSEG(I_P,IL)=J
           END IF
           VALID1(I_P,IL,J)=1
           PE(I_P,IL,J)=PE(I_P,IL,ILSPLIT)
           NL1(I_P,IL,J)=NL1(I_P,IL,ILSPLIT)
           NL1(I_P,IL,ILSPLIT)=J
           PS(I_P,IL,J)=OBS(I_P,ISPLIT)
           PE(I_P,IL,ILSPLIT)=OBS(I_P,ISPLIT)
           VALOBS(I_P,ISPLIT)=0
           VALSPLIT=1
           RR1(I_P,IL,J)=RMAX1
           CC1(I_P,IL,J)=CC1(I_P,IL,IS)
           IS=NL1(I_P,IL,J)
         ELSE
           CC1(I_P,IL,IS)=CC(I_P,IL,IS)
           IS=NL1(I_P,IL,IS)
         END IF
         IF (IS<2) EXIT
       END DO

 !CHECK THE ANNIHILATION BETWEEN LOOPS AND OBSTACLES
       IS=1;VALANN=0
       DO
         NN=NL1(I_P,IL,IS)
         ILSPLIT=0
         IF((MAG(CC1(I_P,IL,IS)-CC1(I_P,IL,NN))/=0).AND.(NN/=0))
 +         THEN
           ANG1=180*(1-DACOS((CC1(I_P,IL,IS)-PE(I_P,IL,IS))*
 +           (CC1(I_P,IL,NN)-PS(I_P,IL,NN))/RR1(I_P,IL,IS)
 +           /RR1(I_P,IL,NN))/PI)
           IF (ANG1<CRIT) THEN
             III=1;VEC=PE(I_P,IL,IS)
             PE(I_P,IL,IS)=PE(I_P,IL,NN)
             CALL CEN_CUR(IL,IS)
             CALL MAX_RAD(IL,IS)
             RR1(I_P,IL,IS)=RMAX1
             IF (ILSPLIT/=0) THEN
               PE(I_P,IL,IS)=OBS(I_P,ISPLIT)
               PS(I_P,IL,NN)=PE(I_P,IL,IS)
               RR1(I_P,IL,NN)=RMAX1
               CC1(I_P,IL,NN)=CC1(I_P,IL,IS)
               VALOBS(I_P,ISPLIT)=0
             ELSE
               IF (NN==1) THEN
                 PS(I_P,IL,1)=PS(I_P,IL,NL1(I_P,IL,1))
                 PE(I_P,IL,1)=PE(I_P,IL,NL1(I_P,IL,1))
                 CC(I_P,IL,1)=CC(I_P,IL,NL1(I_P,IL,1))
                 CC1(I_P,IL,1)=CC1(I_P,IL,NL1(I_P,IL,1))
                 RR1(I_P,IL,1)=RR1(I_P,IL,NL1(I_P,IL,1))
                 VALID1(I_P,IL,NL1(I_P,IL,1))=0
                 NL1(I_P,IL,1)=NL1(I_P,IL,NL1(I_P,IL,1))
                 NL1(I_P,IL,IS)=1;CC1(I_P,IL,IS)=CC(I_P,IL,IS)
               ELSE
                 VALID1(I_P,IL,NN)=0
                 NL1(I_P,IL,IS)=NL1(I_P,IL,NN)
                 CC1(I_P,IL,IS)=CC(I_P,IL,IS)
                 !IF (NN==NSEG(I_P,IL)) THEN;NSEG(I_P,IL)=IS;END IF
               END IF
             END IF
             VALANN=1
           END IF
         END IF
         IF (VALID1(I_P,IL,NN)/=0)THEN
           IS=NL1(I_P,IL,IS)
         END IF
         IF ((IS==0).OR.(NL1(I_P,IL,IS)==0).OR.(IS==1)) EXIT
       END DO
```

```
   !DRAW ALL SEGMENTS
         WRITE(*,*)"IL",IL
         IS=1
         DO
           SP=PS(I_P,IL,IS)+ROBS*(CC1(I_P,IL,IS)
  +          -PS(I_P,IL,IS))/RR1(I_P,IL,IS)
           EP=PE(I_P,IL,IS)+ROBS*(CC1(I_P,IL,IS)
  +          -PE(I_P,IL,IS))/RR1(I_P,IL,IS)
           WRITE(5,FMT=15) "ARC ",SP%V(1),",",SP%V(2)," C ",
  +         CC1(I_P,IL,IS)%V(1),",",CC1(I_P,IL,IS)%V(2)," ",
  +         EP%V(1),",",EP%V(2)
15           FORMAT(A4,E12.6,A1,E12.6,A3,E12.6,A1,
  +              E12.6,A1,E12.6,A1,E12.6)

           ANG1=DASIN(MAG(SP-EP)/2/RR1(I_P,IL,IS))*2
           NP=INT(RR1(I_P,IL,IS)*ANG1/AS)
           V1=SP-CC1(I_P,IL,IS)
           WRITE(6,FMT=16)"3DPOLY"
16           FORMAT(A6)
           DO I=0,NP
             CA=DCOS(ANG1*I/NP);SA=DSIN(ANG1*I/NP)
             LV%V(1)=CA*V1%V(1)-SA*V1%V(2)+CC1(I_P,IL,IS)%V(1)
             LV%V(2)=SA*V1%V(1)+CA*V1%V(2)+CC1(I_P,IL,IS)%V(2)
             LV%V(3)=0
             !CALL TRANS1(LV,GV,MILLER(I_P),ORIGIN(I_P),1)
             GV=TRANS(ES(I_P))*LV+ORIGIN(I_P)
             IF (I==NP) THEN
               WRITE(6,FMT=18)GV%V(1),",",GV%V(2),",",GV%V(3)," "
18               FORMAT(E12.6,A1,E12.6,A1,E12.6,A1)
             ELSE
               WRITE(6,FMT=17)GV%V(1),",",GV%V(2),",",GV%V(3)
17               FORMAT(E12.6,A1,E12.6,A1,E12.6)
             END IF
           END DO

           IS=NL1(I_P,IL,IS)
           IF (IS<2) EXIT
         END DO

         IF ((VALANN==0).AND.(VALSPLIT==0)) THEN
           VALSTOP(I_P,IL)=0
           IF (LOOPCHANGE(I_P,IL)==1) THEN
             NLOOP(I_P)=NLOOP(I_P)+1
             N_LOOP=N_LOOP+1
             LOOPTYPE(I_P,NLOOP(I_P))=1
             I1=1
             DO I1=1,NSEG(I_P,IL)
               IF (VALID1(I_P,IL,I1)/=0)THEN
                 I2=I1
               END IF
             END DO
             PL(I_P,NLOOP(I_P),1)=PS(I_P,IL,1)
             PL(I_P,NLOOP(I_P),3)=PE(I_P,IL,I2)
             N_NODE(I_P,NLOOP(I_P))=3
             A1=H*(PL(I_P,NLOOP(I_P),3)-PL(I_P,NLOOP(I_P),1))
             PL(I_P,NLOOP(I_P),2)=PL(I_P,NLOOP(I_P),1)+
  +                           A1-D%V(3)**A1
             VALID(I_P,NLOOP(I_P))=1
           END IF
         ELSE
           VALSTOP1(I_P)=1
         END IF

       END IF
     END DO
   !CHECK THE ANNIHILATION BETWEEN LOOPS
    D1=999
    DO IL=1,NLOOP1(I_P)
      DO JL=IL,NLOOP1(I_P)
        DO I=1,NSEG(I_P,IL)
          DO J=I,NSEG(I_P,JL)
            IF ((VALID1(I_P,IL,I)==1).AND.(VALID1(I_P,JL,J)==1))
+             THEN
              P1=(CC1(I_P,IL,I)-PS(I_P,JL,J))**
+                 (PE(I_P,JL,J)-PS(I_P,JL,J))
              P2=(CC1(I_P,JL,J)-PS(I_P,IL,I))**
+                 (PE(I_P,IL,I)-PS(I_P,IL,I))
              IF ((P1%V(3)>0).AND.(P2%V(3)>0)) THEN
                P1=(PS(I_P,IL,I)-CC1(I_P,IL,I))**
+                   (CC1(I_P,JL,J)-CC1(I_P,IL,I))
                P2=(PE(I_P,IL,I)-CC1(I_P,IL,I))**
+                   (CC1(I_P,JL,J)-CC1(I_P,IL,I))
                P3=(PS(I_P,JL,J)-CC1(I_P,JL,J))**
+                   (CC1(I_P,IL,I)-CC1(I_P,JL,J))
                P4=(PE(I_P,JL,J)-CC1(I_P,JL,J))**
+                   (CC1(I_P,IL,I)-CC1(I_P,JL,J))
                DD=MAG(CC1(I_P,IL,I)-CC1(I_P,JL,J))-
+                  RR1(I_P,IL,I)-RR1(I_P,JL,J)
                IF ((P3%V(3)*P4%V(3)<0).AND.(P1%V(3)*P2%V(3)<0)
+                 .AND.(DD>0).AND.(DD<D1)) THEN
                  D1=DD;IL_ANN=IL;JL_ANN=JL;I_ANN=I;J_ANN=J
                END IF
              END IF
            END IF
          END DO
        END DO
      END DO
    END DO

    IF (D1<30) THEN
      IF (IL_ANN/=JL_ANN)THEN
        CALL ANNIHILATE
      ELSE IF((I_ANN/=NL1(I_P,JL_ANN,J_ANN)).AND.(I_ANN/=
+       NL1(I_P,JL_ANN,NL1(I_P,JL_ANN,J_ANN))).AND.(J_ANN/=
+       NL1(I_P,IL_ANN,I_ANN)).AND.(LOOPTYPE1(I_P,IL_ANN)==1)
+       .AND.(J_ANN/=NL1(I_P,IL_ANN,NL1(I_P,IL_ANN,I_ANN)))) THEN
        WRITE(*,*)"ANNIHILATE"
        CALL ANNIHILATION
      END IF
    END IF

  END IF
END DO

END SUBROUTINE LOOP_OBS

!************************************************************!

      SUBROUTINE MAX_RAD(ILL,IS)
USE VECTORS;USE VARIABLES
INTEGER,INTENT(IN)::ILL,IS
TYPE(VECTOR)::P,P1
DOUBLE PRECISION::RR
INTEGER::I
RR=RMAX1
DO I=1,NOBS(I_P)
  IF (VALOBS(I_P,I)==1) THEN
    IF (MAG(OBS(I_P,I)-CC(I_P,ILL,IS))<RR) THEN
      P=(OBS(I_P,I)-PS(I_P,ILL,IS))**
+       (PE(I_P,ILL,IS)-PS(I_P,ILL,IS))
      IF (III==1) THEN
    P1=(OBS(I_P,I)-PS(I_P,ILL,IS))**(VEC-PS(I_P,ILL,IS))
    IF ((P%V(3)<0).AND.(P1%V(3)>0)) THEN
      VALOBS(I_P,I)=0
    END IF
   END IF
   IF (P%V(3)>0) THEN
    CALL RAD_CUR(ILL,IS,I)
    IF ((RC>RMAX1).AND.(RC<1.0D6))THEN;RMAX1=RC;ILSPLIT=IS
      ISPLIT=I;CC1(I_P,ILL,IS)=V;END IF
   END IF
  END IF
 END IF
END DO
III=0
END SUBROUTINE MAX_RAD

!*************************************************************!


   DOUBLE PRECISION FUNCTION MIN_DIST(IL1,IL2)
USE VARIABLES;USE VECTORS;DOUBLE PRECISION,EXTERNAL::P_C_DIST
INTEGER,INTENT(IN)::IL1,IL2
DOUBLE PRECISION::A,BB,MM1,DMIN,MM
INTEGER::K1,I,J1,J2,M0,TT,I1,I2,I3
DO K1=1,4;M(K1)=999.0D0;END DO
IF (IL1==IL2) THEN;TT=1;ELSE;TT=0;END IF
I1=N_NODE(I_P,IL1)-TT*INT(N_NODE(I_P,IL1)/2+3)
I2=1+TT*INT(N_NODE(I_P,IL2)/2+3)
I3=N_NODE(I_P,IL2)
DO I=1,I1
 MM1=P_C_DIST(PL(I_P,IL1,I),IL2,I2,I3);
 IF (MM1<M(1)) THEN;M(2)=M(1);M(1)=MM1;KN(2)=KN(1);KN(1)=I;M0=MN
 ELSE;IF (MM1<M(2)) THEN;M(2)=MM1;KN(2)=I;END IF;END IF
END DO
DMIN=999.0D0
IF (M(1)<999.0D0) THEN
 IF (ABS(KN(1)-KN(2))>1) THEN
   IF((LOOPTYPE(I_P,IL1)==1).AND.((KN(1)==1).OR.
+     (KN(1)==N_NODE(I_P,IL1))))THEN
    IF (KN(1)==1) THEN;KN(2)=KN(1)+1;ELSE;KN(2)=KN(1)-1;END IF
   ELSE
    A=P_C_DIST(PL(I_P,IL1,KN(1)-1),IL2,I2,I3)
    BB=P_C_DIST(PL(I_P,IL1,KN(1)+1),IL2,I2,I3)
    IF (A<BB) THEN;KN(2)=KN(1)-1;ELSE;KN(2)=KN(1)+1;END IF
   END IF
 END IF
 IF ((LOOPTYPE(I_P,IL2)==1).AND.((M0==1).OR.
+    (M0==N_NODE(I_P,IL2)))) THEN
   KN(3)=M0;IF (M0==1) THEN;KN(4)=M0+1;ELSE;KN(4)=M0-1;END IF
 ELSE
   A=P_C_DIST(PL(I_P,IL2,M0-1),IL1,1,I1)
   BB=P_C_DIST(PL(I_P,IL2,M0+1),IL1,1,I1)
   KN(3)=M0;IF (A<BB) THEN;KN(4)=M0-1;ELSE;KN(4)=M0+1;END IF
 END IF
 IF (KN(1)<KN(2)) THEN;IM(1)=KN(1);ELSE;IM(1)=KN(2);END IF
 IF (KN(3)<KN(4)) THEN;IM(2)=KN(3);ELSE;IM(2)=KN(4);END IF
 DO J1=0,20;CALL PLOT(IL1,IM(1),J1/20.0D0);R1=R_PLOT
   DO J2=0,20;CALL PLOT(IL2,IM(2),J2/20.0D0);MM=MAG(R_PLOT-R1)
    IF (MM<DMIN) THEN;DMIN=MM;UM(1)=J1/20.0;UM(2)=J2/20.0;END IF
   END DO
 END DO
END IF
WRITE(*,*)"IL1,IL2,KN(1),KN(3),DMIN",IL1,IL2,KN(1),KN(3),DMIN
MIN_DIST=DMIN
END FUNCTION MIN_DIST
```

```
!**************************************************************!

     SUBROUTINE OPEN_IO_FILES

OPEN(UNIT=1,FILE="MATERIAL_INPUT.TXT",ACTION="READ")
OPEN(UNIT=2,FILE="GEOMETRY_INPUT.TXT")
OPEN(UNIT=3,FILE="INTERACTION_GEOM_INPUT.TXT",ACTION="READ")
OPEN(UNIT=4,FILE="FIELD_INPUT.TXT")        !!RUDY 3_1
!OPEN(UNIT=5,FILE="FEM_INPUT.TXT",ACTION="READ")
OPEN(UNIT=6,FILE="OBSTACLE_COOR_INPUT.TXT",ACTION="READ")
OPEN(UNIT=7,FILE="COORD_AND_FEM_STRESS_INPUT.TXT",ACTION="READ") !!RUDY 3_6
OPEN(UNIT=8,FILE="DYNAMIC_LOOP_DATA.TXT",ACTION="READ")!!RUDY 5/5

OPEN(UNIT=11,FILE="AUTOCAD_OUTPUT.SCR")
!OPEN(UNIT=12,FILE="DISL_GEOM_OUTPUT.TXT")
OPEN(UNIT=13,FILE="INTERACTION_OUTPUT.TXT")
OPEN(UNIT=14,FILE="FEM_OUTPUT.TXT")
OPEN(UNIT=15,FILE="ELASTIC_FIELD_OUTPUT.TXT")
!OPEN(UNIT=16,FILE="LOOP_UPDATE.TXT")

OPEN(UNIT=20,FILE="RESULTANT_F.TXT")
OPEN(UNIT=21,FILE="IMAGE_F.TXT")
OPEN(UNIT=22,FILE="PEIRELS_F.TXT")
OPEN(UNIT=23,FILE="APPLIED_F.TXT")
OPEN(UNIT=24,FILE="SELF_F.TXT")
END SUBROUTINE OPEN_IO_FILES

!**************************************************************!

     DOUBLE PRECISION FUNCTION P_C_DIST(P,ILL,I1,I2)
USE VARIABLES;USE VECTORS
TYPE(VECTOR),INTENT(IN)::P;INTEGER,INTENT(IN)::ILL,I1,I2
DOUBLE PRECISION::DOT1,DOT2,M0,MM1,UU;INTEGER::K
MM1=999.0D0
MN=0.0D0
DO K=I1,I2-LOOPTYPE(I_P,ILL)
  DOT1=(PL(I_P,ILL,K)-P)*TL(I_P,ILL,K)
  DOT2=(PL(I_P,ILL,K+1)-P)*TL(I_P,ILL,K+1)
  IF (DOT1*DOT2<0) THEN
    U=DABS(DOT1)/(DABS(DOT1)+DABS(DOT2))
    CALL PLOT(ILL,K,UU)
    M0=MAG(P-R_PLOT)
    IF (M0<MM1) THEN
      MM1=M0
      IF (DABS(DOT1)>DABS(DOT2)) THEN;MN=K+1;ELSE;MN=K;END IF
    END IF
  END IF
END DO
P_C_DIST=MM1
END FUNCTION P_C_DIST

!**************************************************************!

     SUBROUTINE PLOT(ILL,I,UU)
USE VECTORS;USE VARIABLES
INTEGER,INTENT(IN)::ILL,I;DOUBLE PRECISION,INTENT(IN)::UU
DOUBLE PRECISION::C1,C2,C3,C4,C5,C6,U3,U4,U5
SELECT CASE(5)
CASE(1)
  R_PLOT=(1-UU)*PL(I_P,ILL,I)+UU*PL(I_P,ILL,I)
CASE(3)
  U3=UU*UU*UU
  C1=2*U3-3*UU*UU+1
  C2=1-C1
  C3=U3-2*UU*UU+UU
  C4=U3-UU*UU
  R_PLOT=C1*PL(I_P,ILL,I)+C2*PL(I_P,ILL,I+1)+C3*TL(I_P,ILL,I)+
+        C4*TL(I_P,ILL,I+1)
 CASE(5)
  U3=UU*UU*UU
  U4=U3*UU
  U5=U4*UU
  C1=-6*U5+15*U4-10*U3+1
  C2=6*U5-15*U4+10*U3
  C3=-3*U5+8*U4-6*U3+UU
  C4=-3*U5+7*U4-4*U3
  C5=-0.5*U5+1.5*U4-1.5*U3+0.5*UU*UU
  C6=0.5*U5-U4+0.5*U3
  R_PLOT=C1*PL(I_P,ILL,I)+C2*PL(I_P,ILL,I+1)+C3*T2(I_P,ILL,I)*
+        TL(I_P,ILL,I)+C4*T1(I_P,ILL,I+1)*TL(I_P,ILL,I+1)+
+        C5*N2(I_P,ILL,I)*NL(I_P,ILL,I)+C6*N1(I_P,ILL,I+1)*
+        NL(I_P,ILL,I+1)

 END SELECT
 END SUBROUTINE PLOT

!*************************************************************!

           SUBROUTINE quadrature(i,pos,wt)

 intent(in)::i
 intent(out)::pos,wt
 integer::i,j
 double precision,dimension(300)::pos,wt

 if (i>64) then
    j=128
 else if (i>32) then
    j=64
 else if (i>16) then
    j=32
 else if (i<2) then
    j=2
 else
    j=i
 end if

 select case (j)

           case (2)

            pos(1)= -0.577350269189626;    wt(1)= 1.000000000000000
            pos(2)=  0.577350269189626;    wt(2)= 1.000000000000000

           case (3)

            pos(1)=-0.774596669241483;     wt(1)=       0.555555555555556
            pos(2)= 0.000000000000000;     wt(2)=       0.888888888888889
            pos(3)= 0.774596669241483;     wt(3)=  0.555555555555556

           case(4)

            pos(1)=-0.861136311594053;     wt(1)=       0.347854845137454
            pos(2)=-0.339981043584856;     wt(1)=       0.652145154862546
            pos(3)= 0.339981043584856;     wt(3)=       0.652145154862546
            pos(4)= 0.861136311594053;     wt(4)=       0.347854845137454

           case(5)

            pos(1)=-0.906179845938664;     wt(1)=       0.236926885056189
            pos(2)=-0.538469310105683;     wt(2)=       0.478628670499366
            pos(3)= 0.000000000000000;     wt(3)=       0.568888888888889
 pos(4)= 0.538469310105683;    wt(4)=      0.478628670499366
 pos(5)= 0.906179845938664;    wt(5)=      0.236926885056189

case(6)

 pos(1)=-0.932469514203152;    wt(1)=      0.171324492379170
 pos(2)=-0.661209386466265;    wt(2)=      0.360761573048139
 pos(3)=-0.238619186083197;    wt(3)=      0.467913934572691
 pos(4)= 0.238619186083197;    wt(4)=      0.467913934572691
 pos(5)= 0.661209386466265;    wt(5)=      0.360761573048139
 pos(6)= 0.932469514203152;    wt(6)=      0.171324492379170

case(7)

 pos(1)=-0.949107912342759;    wt(1)=      0.129484966168870
 pos(2)=-0.741531185599394;    wt(2)=      0.279705391489277
 pos(3)=-0.405845151377397;    wt(3)=      0.381830050505119
 pos(4)= 0.000000000000000;    wt(4)=      0.417959183673469
 pos(5)= 0.405845151377397;    wt(5)=      0.381830050505119
 pos(6)= 0.741531185599394;    wt(6)=      0.279705391489277
 pos(7)= 0.949107912342759;    wt(7)=      0.129484966168870

case(8)

 pos(1)=-0.960289856497536;    wt(1)=      0.101228536290376
 pos(2)=-0.796666477413627;    wt(2)=      0.222381034453374
 pos(3)=-0.525532409916329;    wt(3)=      0.313706645877887
 pos(4)=-0.183434642495650;    wt(4)=      0.362683783378362
 pos(5)= 0.183434642495650;    wt(5)=      0.362683783378362
 pos(6)= 0.525532409916329;    wt(6)=      0.313706645877887
 pos(7)= 0.796666477413627;    wt(7)=      0.222381034453374
 pos(8)= 0.960289856497536;    wt(8)=      0.101228536290376

case (9)

 pos(1)=-0.968160239507626;    wt(1)=      0.081274388361574
 pos(2)=-0.836031107326636;    wt(2)=      0.180648160694857
 pos(3)=-0.613371432700590;    wt(3)=      0.260610696402935
 pos(4)=-0.324253423403809;    wt(4)=      0.312347077040003
 pos(5)= 0.000000000000000;    wt(5)=      0.330239355001260
 pos(6)= 0.324253423403809;    wt(6)=      0.312347077040003
 pos(7)= 0.613371432700590;    wt(7)=      0.260610696402935
 pos(8)= 0.836031107326636;    wt(8)=      0.180648160694857
 pos(9)= 0.968160239507626;    wt(9)=      0.081274388361574

case (10)

 pos(1)=-0.973906528517172;    wt(1)=      0.066671344308668
 pos(2)=-0.865063366688985;    wt(2)=      0.149451349150581
 pos(3)=-0.679409568299024;    wt(3)=      0.219086362515982
 pos(4)=-0.433395394129247;    wt(4)=      0.269266719309996
 pos(5)=-0.148874338981631;    wt(5)=      0.295524224714753
 pos(6)= 0.148874338981631;    wt(6)=      0.295524224714753
 pos(7)= 0.433395394129247;    wt(7)=      0.269266719309996
 pos(8)= 0.679409568299024;    wt(8)=      0.219086362515982
 pos(9)= 0.865063366688985;    wt(9)=      0.149451349150581
 pos(10)=0.973906528517172;    wt(10)=     0.066671344308668


case (11)

 pos(1)=-0.978228658146057;    wt(1)=      0.055668567116174
 pos(2)=-0.887062599768095;    wt(2)=      0.125580369464905
 pos(3)=-0.730152005574049;    wt(3)=      0.186290210927734
 pos(4)=-0.519096129206812;    wt(4)=      0.233193764591990
 pos(5)=-0.269543155952345;    wt(5)=      0.262804544510247
 pos(6)= 0.000000000000000;     wt(6)=      0.272925086777901
 pos(7)= 0.269543155952345;     wt(7)=      0.262804544510247
 pos(8)= 0.519096129206812;     wt(8)=      0.233193764591990
 pos(9)= 0.730152005574049;     wt(9)=      0.186290210927734
 pos(10)=0.887062599768095;     wt(10)=     0.125580369464905
 pos(11)=0.978228658146057;     wt(11)=     0.055668567116174

case (12)

 pos(1)=-0.981560634246719;     wt(1)=      0.047175336386512
 pos(2)=-0.904117256370475;     wt(2)=      0.106939325995318
 pos(3)=-0.769902674194305;     wt(3)=      0.160078328543346
 pos(4)=-0.587317954286617;     wt(4)=      0.203167426723066
 pos(5)=-0.367831498998180;     wt(5)=      0.233492536538355
 pos(6)=-0.125233408511469;     wt(6)=      0.249147045813403
 pos(7)= 0.125233408511469;     wt(7)=      0.249147045813403
 pos(8)= 0.367831498998180;     wt(8)=      0.233492536538355
 pos(9)= 0.587317954286617;     wt(9)=      0.203167426723066
 pos(10)=0.769902674194305;     wt(10)=     0.160078328543346
 pos(11)=0.904117256370475;     wt(11)=     0.106939325995318
 pos(12)=0.981560634246719;     wt(12)=     0.047175336386512

case (13)

 pos(1)=-0.984183054718588;     wt(1)=      0.040484004765316
 pos(2)=-0.917598392222975;     wt(2)=      0.092121498837728
 pos(3)=-0.801578090733310;     wt(3)=      0.138873510219787
 pos(4)=-0.642349339440340;     wt(4)=      0.178145980761946
 pos(5)=-0.448492751036447;     wt(5)=      0.207816047536889
 pos(6)=-0.230458315955135;     wt(6)=      0.226283180262897
 pos(7)= 0.000000000000000;     wt(7)=      0.232551553230874
 pos(8)= 0.230458315955135;     wt(8)=      0.226283180262897
 pos(9)= 0.448492751036447;     wt(9)=      0.207816047536889
 pos(10)=0.642349339440340;     wt(10)=     0.178145980761946
 pos(11)=0.801578090733310;     wt(11)=     0.138873510219787
 pos(12)=0.917598392222975;     wt(12)=     0.092121498837728
 pos(13)=0.984183054718588;     wt(13)=     0.040484004765316

case (14)

 pos(1)=-0.986283808696812;     wt(1)=      0.035119460331752
 pos(2)=-0.928434883663574;     wt(2)=      0.080158087159760
 pos(3)=-0.827201315069765;     wt(3)=      0.121518570687903
 pos(4)=-0.687292904811685;     wt(4)=      0.157203167158194
 pos(5)=-0.515248636358154;     wt(5)=      0.185538397477938
 pos(6)=-0.319112368927890;     wt(6)=      0.205198463721296
 pos(7)=-0.108054948707344;     wt(7)=      0.215263853463158
 pos(8)= 0.108054948707344;     wt(8)=      0.215263853463158
 pos(9)= 0.319112368927890;     wt(9)=      0.205198463721296
 pos(10)=0.515248636358154;     wt(10)=     0.185538397477938
 pos(11)=0.687292904811685;     wt(11)=     0.157203167158194
 pos(12)=0.827201315069765;     wt(12)=     0.121518570687903
 pos(13)=0.928434883663574;     wt(13)=     0.080158087159760
 pos(14)=0.986283808696812;     wt(14)=     0.035119460331752

case (15)

 pos(1)=-0.987992518020485;     wt(1)=      0.030753241996117
 pos(2)=-0.937273392400706;     wt(2)=      0.070366047488108
 pos(3)=-0.848206583410427;     wt(3)=      0.107159220467172
 pos(4)=-0.724417731360170;     wt(4)=      0.139570677926154
 pos(5)=-0.570972172608539;     wt(5)=      0.166269205816994
 pos(6)=-0.394151347077563;     wt(6)=      0.186161000015562
 pos(7)=-0.201194093997435;     wt(7)=      0.198431485327112
 pos(8)= 0.000000000000000;     wt(8)=      0.202578241925561
 pos(9)= 0.201194093997435;     wt(9)=      0.198431485327112
```

```
        pos(10)=0.394151347077563;    wt(10)=    0.186161000015562
        pos(11)=0.570972172608539;    wt(11)=    0.166269205816994
        pos(12)=0.724417731360170;    wt(12)=    0.139570677926154
        pos(13)=0.848206583410427;    wt(13)=    0.107159220467172
        pos(14)=0.937273392400706;    wt(14)=    0.070366047488108
  pos(15)=0.987992518020485;          wt(15)=    0.030753241996117

       case (16)

        pos(1)=-0.989400934991650;    wt(1)=     0.027152459411754
        pos(2)=-0.944575023073233;    wt(2)=     0.062253523938648
        pos(3)=-0.865631202387832;    wt(3)=     0.095158511682493
        pos(4)=-0.755404408355003;    wt(4)=     0.124628971255534
        pos(5)=-0.617876244402644;    wt(5)=     0.149595988816577
        pos(6)=-0.458016777657227;    wt(6)=     0.169156519395003
        pos(7)=-0.281603550779259;    wt(7)=     0.182603415044924
        pos(8)=-0.095012509837637;    wt(8)=     0.189450610455068
        pos(9)= 0.095012509837637;    wt(9)=     0.189450610455068
        pos(10)=0.281603550779259;    wt(10)=    0.182603415044924
        pos(11)=0.458016777657227;    wt(11)=    0.169156519395003
        pos(12)=0.617876244402644;    wt(12)=    0.149595988816577
        pos(13)=0.755404408355003;    wt(13)=    0.124628971255534
        pos(14)=0.865631202387832;    wt(14)=    0.095158511682493
        pos(15)=0.944575023073233;    wt(15)=    0.062253523938648
        pos(16)=0.989400934991650;    wt(16)=    0.027152459411754

case (32)

pos( 1)=  -.997263861849482;   wt( 1)=  .007018610009470
pos( 2)=  -.985611511545269;   wt( 2)=  .016274394730906
pos( 3)=  -.964762255587506;   wt( 3)=  .025392065309262
pos( 4)=  -.934906075937740;   wt( 4)=  .034273862913020
pos( 5)=  -.896321155766052;   wt( 5)=  .042835898022227
pos( 6)=  -.849367613732570;   wt( 6)=  .050998059262376
pos( 7)=  -.794483795967942;   wt( 7)=  .058684093478535
pos( 8)=  -.732182118740290;   wt( 8)=  .065822222776362
pos( 9)=  -.663044266930215;   wt( 9)=  .072345794108850
pos(10)=  -.587715757240762;   wt(10)=  .078193895787070
pos(11)=  -.506899908932229;   wt(11)=  .083311924226945
pos(12)=  -.421351276130636;   wt(12)=  .087652093004404
pos(13)=  -.331868602282128;   wt(13)=  .091173878695764
pos(14)=  -.239287362252137;   wt(14)=  .093844399080805
pos(15)=  -.144471961582796;   wt(15)=  .095638720079274
pos(16)=  -.048307665687738;   wt(16)=  .096540088514728
pos(17)=   .048307665687738;   wt(17)=  .096540088514728
pos(18)=   .144471961582797;   wt(18)=  .095638720079275
pos(19)=   .239287362252137;   wt(19)=  .093844399080805
pos(20)=   .331868602282128;   wt(20)=  .091173878695764
pos(21)=   .421351276130635;   wt(21)=  .087652093004403
pos(22)=   .506899908932229;   wt(22)=  .083311924226947
pos(23)=   .587715757240762;   wt(23)=  .078193895787070
pos(24)=   .663044266930215;   wt(24)=  .072345794108848
pos(25)=   .732182118740289;   wt(25)=  .065822222776362
pos(26)=   .794483795967942;   wt(26)=  .058684093478536
pos(27)=   .849367613732570;   wt(27)=  .050998059262375
pos(28)=   .896321155766052;   wt(28)=  .042835898022227
pos(29)=   .934906075937740;   wt(29)=  .034273862913022
pos(30)=   .964762255587506;   wt(30)=  .025392065309262
pos(31)=   .985611511545268;   wt(31)=  .016274394730905
pos(32)=   .997263861849481;   wt(32)=  .007018610009470

case (64)

pos( 1)=  -.999305041735772;   wt( 1)=  .001783280721697
pos( 2)=  -.996340116771955;   wt( 2)=  .004147033260562
pos( 3)=  -.991013371476744;   wt( 3)=  .006504457968979
```

```
 pos( 4)=  -.983336253884626;    wt( 4)=  .008846759826364
 pos( 5)=  -.973326827789911;    wt( 5)=  .011168139460131
 pos( 6)=  -.961008799652054;    wt( 6)=  .013463047896719
 pos( 7)=  -.946411374858403;    wt( 7)=  .015726030476023
 pos( 8)=  -.929569172131939;    wt( 8)=  .017951715775697
 pos( 9)=  -.910522137078502;    wt( 9)=  .020134823153531
 pos(10)=  -.889315445995114;    wt(10)=   .022270173808383
 pos(11)=  -.865999398154093;    wt(11)=   .024352702568711
 pos(12)=  -.840629296252580;    wt(12)=   .026377469715055
 pos(13)=  -.813265315122797;    wt(13)=   .028339672614259
 pos(14)=  -.783972358943341;    wt(14)=   .030234657072403
 pos(15)=  -.752819907260532;    wt(15)=   .032057928354852
 pos(16)=  -.719881850171611;    wt(16)=   .033805161837141
 pos(17)=  -.685236313054233;    wt(17)=   .035472213256883
 pos(18)=  -.648965471254657;    wt(18)=   .037055128540240
 pos(19)=  -.611155355172394;    wt(19)=   .038550153178615
 pos(20)=  -.571895646202634;    wt(20)=   .039953741132720
 pos(21)=  -.531279464019895;    wt(21)=   .041262563242625
 pos(22)=  -.489403145707052;    wt(22)=   .042473515123654
 pos(23)=  -.446366017253464;    wt(23)=   .043583724529323
 pos(24)=  -.402270157963991;    wt(24)=   .044590558163757
 pos(25)=  -.357220158337668;    wt(25)=   .045491627927418
 pos(26)=  -.311322871990211;    wt(26)=   .046284796581314
 pos(27)=  -.264687162208767;    wt(27)=   .046968182816209
 pos(28)=  -.217423643740007;    wt(28)=   .047540165714831
 pos(29)=  -.169644420423993;    wt(29)=   .047999388596458
 pos(30)=  -.121462819296121;    wt(30)=   .048344762234804
 pos(31)=  -.072993121787799;    wt(31)=   .048575467441504
 pos(32)=  -.024350292663424;    wt(32)=   .048690957009140
 pos(33)=   .024350292663425;    wt(33)=   .048690957009139
 pos(34)=   .072993121787799;    wt(34)=   .048575467441503
 pos(35)=   .121462819296120;    wt(35)=   .048344762234803
 pos(36)=   .169644420423993;    wt(36)=   .047999388596458
 pos(37)=   .217423643740007;    wt(37)=   .047540165714831
 pos(38)=   .264687162208767;    wt(38)=   .046968182816210
 pos(39)=   .311322871990211;    wt(39)=   .046284796581314
 pos(40)=   .357220158337668;    wt(40)=   .045491627927417
 pos(41)=   .402270157963991;    wt(41)=   .044590558163758
 pos(42)=   .446366017253464;    wt(42)=   .043583724529324
 pos(43)=   .489403145707053;    wt(43)=   .042473515123653
 pos(44)=   .531279464019895;    wt(44)=   .041262563242624
 pos(45)=   .571895646202634;    wt(45)=   .039953741132721
 pos(46)=   .611155355172393;    wt(46)=   .038550153178616
 pos(47)=   .648965471254657;    wt(47)=   .037055128540241
 pos(48)=   .685236313054233;    wt(48)=   .035472213256881
 pos(49)=   .719881850171611;    wt(49)=   .033805161837142
 pos(50)=   .752819907260532;    wt(50)=   .032057928354852
 pos(51)=   .783972358943342;    wt(51)=   .030234657072402
 pos(52)=   .813265315122797;    wt(52)=   .028339672614259
 pos(53)=   .840629296252580;    wt(53)=   .026377469715055
 pos(54)=   .865999398154092;    wt(54)=   .024352702568711
 pos(55)=   .889315445995114;    wt(55)=   .022270173808382
 pos(56)=   .910522137078503;    wt(56)=   .020134823153530
 pos(57)=   .929569172131940;    wt(57)=   .017951715775697
 pos(58)=   .946411374858403;    wt(58)=   .015726030476025
 pos(59)=   .961008799652054;    wt(59)=   .013463047896719
 pos(60)=   .973326827789911;    wt(60)=   .011168139460130
 pos(61)=   .983336253884626;    wt(61)=   .008846759826364
 pos(62)=   .991013371476744;    wt(62)=   .006504457968979
 pos(63)=   .996340116771955;    wt(63)=   .004147033260562
 pos(64)=   .999305041735772;    wt(64)=   .001783280721697

case (128)

 pos(  1)=  -.999824887947132;    wt(  1)=   .000449380960291
 pos(  2)=  -.999077459977377;    wt(  2)=   .001045812679341
pos(  3)=  -.997733248625515;    wt(  3)=   .001642503018669
pos(  4)=  -.995792758534981;    wt(  4)=   .002238288430963
pos(  5)=  -.993257112900213;    wt(  5)=   .002832751471457
pos(  6)=  -.990127818491735;    wt(  6)=   .003425526040911
pos(  7)=  -.986406742724587;    wt(  7)=   .004016254983738
pos(  8)=  -.982096108435719;    wt(  8)=   .004604584256703
pos(  9)=  -.977198491463908;    wt(  9)=   .005190161832676
pos( 10)=  -.971716818747136;    wt( 10)=   .005772637542867
pos( 11)=  -.965654366431966;    wt( 11)=   .006351663161706
pos( 12)=  -.959014757853700;    wt( 12)=   .006926892566899
pos( 13)=  -.951801961341265;    wt( 13)=   .007497981925635
pos( 14)=  -.944020287830221;    wt( 14)=   .008064589890485
pos( 15)=  -.935674388277917;    wt( 15)=   .008626377798618
pos( 16)=  -.926769250878948;    wt( 16)=   .009183009871660
pos( 17)=  -.917310198080961;    wt( 17)=   .009734153415007
pos( 18)=  -.907302883401757;    wt( 18)=   .010279479015831
pos( 19)=  -.896753288049158;    wt( 19)=   .010818660739503
pos( 20)=  -.885667717345397;    wt( 20)=   .011351376324079
pos( 21)=  -.874052796958032;    wt( 21)=   .011877307372741
pos( 22)=  -.861915468939548;    wt( 22)=   .012396139543952
pos( 23)=  -.849262987577969;    wt( 23)=   .012907562739267
pos( 24)=  -.836102915060907;    wt( 24)=   .013411271288616
pos( 25)=  -.822443116955644;    wt( 25)=   .013906964132952
pos( 26)=  -.808291757507913;    wt( 26)=   .014394345004168
pos( 27)=  -.793657294762193;    wt( 27)=   .014873122602147
pos( 28)=  -.778548475506411;    wt( 28)=   .015343010768865
pos( 29)=  -.762974330044094;    wt( 29)=   .015803728659399
pos( 30)=  -.746944166797062;    wt( 30)=   .016255000909785
pos( 31)=  -.730467566741909;    wt( 31)=   .016696557801589
pos( 32)=  -.713554377683587;    wt( 32)=   .017128135423111
pos( 33)=  -.696214708369514;    wt( 33)=   .017549475827118
pos( 34)=  -.678458922447719;    wt( 34)=   .017960327185008
pos( 35)=  -.660297632272646;    wt( 35)=   .018360443937331
pos( 36)=  -.641741692562308;    wt( 36)=   .018749586940545
pos( 37)=  -.622802193910585;    wt( 37)=   .019127523609951
pos( 38)=  -.603490456158549;    wt( 38)=   .019494028058706
pos( 39)=  -.583818021628764;    wt( 39)=   .019848881232830
pos( 40)=  -.563796648226618;    wt( 40)=   .020191871042132
pos( 41)=  -.543438302412811;    wt( 41)=   .020522792486959
pos( 42)=  -.522755152051176;    wt( 42)=   .020841447780752
pos( 43)=  -.501759559136145;    wt( 43)=   .021147646468221
pos( 44)=  -.480464072404172;    wt( 44)=   .021441205539209
pos( 45)=  -.458881419833553;    wt( 45)=   .021721949538052
pos( 46)=  -.437024501037105;    wt( 46)=   .021989710668461
pos( 47)=  -.414906379552274;    wt( 47)=   .022244328893799
pos( 48)=  -.392540275033268;    wt( 48)=   .022485652032746
pos( 49)=  -.369939555349859;    wt( 49)=   .022713535850235
pos( 50)=  -.347117728597636;    wt( 50)=   .022927844143688
pos( 51)=  -.324088435024413;    wt( 51)=   .023128448824387
pos( 52)=  -.300865438877677;    wt( 52)=   .023315229994063
pos( 53)=  -.277462620177904;    wt( 53)=   .023488076016536
pos( 54)=  -.253893966422694;    wt( 54)=   .023646883584448
pos( 55)=  -.230173564226660;    wt( 55)=   .023791557781003
pos( 56)=  -.206315590902079;    wt( 56)=   .023922012136704
pos( 57)=  -.182334305985337;    wt( 57)=   .024038168681024
pos( 58)=  -.158244042714225;    wt( 58)=   .024139957989019
pos( 59)=  -.134059199461188;    wt( 59)=   .024227319222816
pos( 60)=  -.109794231127643;    wt( 60)=   .024300200167972
pos( 61)=  -.085463640504515;    wt( 61)=   .024358557264690
pos( 62)=  -.061081969604139;    wt( 62)=   .024402355633849
pos( 63)=  -.036663790968733;    wt( 63)=   .024431569097850
pos( 64)=  -.012223698960616;    wt( 64)=   .024446180196261
pos( 65)=   .012223698960616;    wt( 65)=   .024446180196263
pos( 66)=   .036663790968734;    wt( 66)=   .024431569097849
pos( 67)=   .061081969604140;    wt( 67)=   .024402355633850
pos( 68)=   .085463640504516;    wt( 68)=   .024358557264692
```

```
  pos( 69)=   .109794231127644;    wt( 69)=   .024300200167971
  pos( 70)=   .134059199461188;    wt( 70)=   .024227319222816
  pos( 71)=   .158244042714225;    wt( 71)=   .024139957989019
  pos( 72)=   .182334305985337;    wt( 72)=   .024038168681024
  pos( 73)=   .206315590902079;    wt( 73)=   .023922012136703
  pos( 74)=   .230173564226660;    wt( 74)=   .023791557781003
  pos( 75)=   .253893966422694;    wt( 75)=   .023646883584448
  pos( 76)=   .277462620177904;    wt( 76)=   .023488076016536
  pos( 77)=   .300865438877677;    wt( 77)=   .023315229994063
  pos( 78)=   .324088435024414;    wt( 78)=   .023128448824387
  pos( 79)=   .347117728597636;    wt( 79)=   .022927844143687
  pos( 80)=   .369939555349859;    wt( 80)=   .022713535850236
  pos( 81)=   .392540275033268;    wt( 81)=   .022485652032746
  pos( 82)=   .414906379552275;    wt( 82)=   .022244328893799
  pos( 83)=   .437024501037104;    wt( 83)=   .021989710668461
  pos( 84)=   .458881419833552;    wt( 84)=   .021721949538052
  pos( 85)=   .480464072404172;    wt( 85)=   .021441205539208
  pos( 86)=   .501759559136144;    wt( 86)=   .021147646468223
  pos( 87)=   .522755152051176;    wt( 87)=   .020841447780751
  pos( 88)=   .543438302412810;    wt( 88)=   .020522792486962
  pos( 89)=   .563796648226618;    wt( 89)=   .020191871042131
  pos( 90)=   .583818021628763;    wt( 90)=   .019848881232830
  pos( 91)=   .603490456158548;    wt( 91)=   .019494028058707
  pos( 92)=   .622802193910585;    wt( 92)=   .019127523609950
  pos( 93)=   .641741692562307;    wt( 93)=   .018749586940545
  pos( 94)=   .660297632272646;    wt( 94)=   .018360443937330
  pos( 95)=   .678458922447719;    wt( 95)=   .017960327185009
  pos( 96)=   .696214708369514;    wt( 96)=   .017549475827118
  pos( 97)=   .713554377683587;    wt( 97)=   .017128135423111
  pos( 98)=   .730467566741908;    wt( 98)=   .016696557801589
  pos( 99)=   .746944166797062;    wt( 99)=   .016255000909785
  pos(100)=   .762974330044094;    wt(100)=   .015803728659399
  pos(101)=   .778548475506412;    wt(101)=   .015343010768867
  pos(102)=   .793657294762193;    wt(102)=   .014873122602146
  pos(103)=   .808291757507914;    wt(103)=   .014394345004167
  pos(104)=   .822443116955644;    wt(104)=   .013906964132951
  pos(105)=   .836102915060907;    wt(105)=   .013411271288616
  pos(106)=   .849262987577969;    wt(106)=   .012907562739268
  pos(107)=   .861915468939549;    wt(107)=   .012396139543951
  pos(108)=   .874052796958032;    wt(108)=   .011877307372740
  pos(109)=   .885667717345397;    wt(109)=   .011351376324080
  pos(110)=   .896753288049158;    wt(110)=   .010818660739504
  pos(111)=   .907302883401757;    wt(111)=   .010279479015832
  pos(112)=   .917310198080960;    wt(112)=   .009734153415007
  pos(113)=   .926769250878948;    wt(113)=   .009183009871662
  pos(114)=   .935674388277916;    wt(114)=   .008626377798617
  pos(115)=   .944020287830220;    wt(115)=   .008064589890486
  pos(116)=   .951801961341264;    wt(116)=   .007497981925634
  pos(117)=   .959014757853700;    wt(117)=   .006926892566899
  pos(118)=   .965654366431965;    wt(118)=   .006351663161707
  pos(119)=   .971716818747137;    wt(119)=   .005772637542865
  pos(120)=   .977198491463907;    wt(120)=   .005190161832677
  pos(121)=   .982096108435719;    wt(121)=   .004604584256702
  pos(122)=   .986406742724586;    wt(122)=   .004016254983739
  pos(123)=   .990127818491734;    wt(123)=   .003425526040910
  pos(124)=   .993257112900213;    wt(124)=   .002832751471459
  pos(125)=   .995792758534981;    wt(125)=   .002238288430962
  pos(126)=   .997733248625514;    wt(126)=   .001642503018669
  pos(127)=   .999077459977376;    wt(127)=   .001045812679341
  pos(128)=   .999824887947132;    wt(128)=   .000449380960292

        end select

END SUBROUTINE quadrature

!***************************************************************!
```

```
      SUBROUTINE RAD_CUR(ILL,IS,I)
   USE VECTORS;USE VARIABLES
   INTEGER,INTENT(IN)::ILL,IS,I
   DOUBLE PRECISION::S
   TYPE(VECTOR)::V21,V23,V1V2,V13,P
   V21=PS(I_P,ILL,IS)-OBS(I_P,I)
   V23=PE(I_P,ILL,IS)-OBS(I_P,I);P=V21**V23
   V1V2=H*(PS(I_P,ILL,IS)+OBS(I_P,I))
   V13=PE(I_P,ILL,IS)-PS(I_P,ILL,IS)
   S=(V13*V23)/(P%V(3))/2
   V=V1V2+S*(D%V(3)**V21);RC=MAG(OBS(I_P,I)-V)
   END SUBROUTINE RAD_CUR

   !**************************************************************!

      SUBROUTINE REFINE(IL1,IL2)
   USE VECTORS;USE VARIABLES
   INTEGER,INTENT(IN)::IL1,IL2
   INTEGER::K1,I,J1,I1,I2,J
   DO K1=2,1,-1
    IF (K1==1) THEN;I=IL1;ELSE;I=IL2;END IF
    IF ((UM(K1)/=0.0D0).AND.(UM(K1)/=1.0D0)) THEN
     CALL PLOT(I,IM(K1),UM(K1));CALL TANG(I,IM(K1),UM(K1))
     DO J1=N_NODE(I_P,I),IM(K1)+1,-1
      CUSP(I_P,I,J1+1)=CUSP(I_P,I,J1);PL(I_P,I,J1+1)=PL(I_P,I,J1)
      TL(I_P,I,J1+1)=TL(I_P,I,J1)
     END DO
     CUSP(I_P,I,IM(K1)+1)=0;PL(I_P,I,IM(K1)+1)=R_PLOT
     TL(I_P,I,IM(K1)+1)=RT/MAG(RT)
     N_NODE(I_P,I)=N_NODE(I_P,I)+1
    END IF
   END DO
   CALL TAN_NOR(IL1);CALL TAN_NOR(IL2)
   WRITE(*,*)"REFINE",IL1,IL2

   END SUBROUTINE REFINE

   !**************************************************************!

      DOUBLE PRECISION FUNCTION SELF_DELTA(CALPHA,KAPPA)
   USE VECTORS;USE VARIABLES
   DOUBLE PRECISION, INTENT(IN)::CALPHA,KAPPA
   DOUBLE PRECISION::C1,C2,C3
   C1=((1-NU*CALPHA**2)/4/PI/(1-NU)+NU*(2*CALPHA**2-1)/2/PI/(1-NU))
  +    *DLOG(8/KAPPA)
   C2=(21+CALPHA**2)/64/PI+(2*CALPHA**2-1)/2/PI
   SELF_DELTA=MS*(C1-C2)*KAPPA*MAG(BURGERS(I_P,IL))
   END FUNCTION SELF_DELTA

   !**************************************************************!
      SUBROUTINE SINGLE_EQUILIBRIUM(I_P1,IL1)
   USE VARIABLES
   USE VECTORS
   DOUBLE PRECISION::CT,ST,DT,LL,PLL
   INTEGER,INTENT(IN)::I_P1,IL1
   TYPE(VECTOR)::NORV,DANG,NORV1,T0,QS,QE,Q,F3
   TYPE(MATRIX)::F1,F2
   INTEGER::I,J

   QS=PL(I_P1,IL1,1)
   QE=PL(I_P1,IL1,3)

   DT=PI/180
   T0=UV(QE-QS)
  SIG=ZERO
  Q=PL(I_P1,IL1,2)
  DO IP_LOC=1,N_PLANE
    DO IL_LOC=1,NLOOP(IP_LOC)
      IF (((IP_LOC/=I_P1).OR.(IL_LOC/=IL1)).AND.
+        ((PROP_DIPOLE(I_P1,IL1,2)/=IP_LOC).OR.
+        (PROP_DIPOLE(I_P1,IL1,3)/=IL_LOC))) THEN
        DO ID_NODE_L=1,N_NODE(IP_LOC,IL_LOC)
          CALL LINE_INTEGRAL(Q,F1,F2,F3,IP_LOC,IL_LOC,ID_NODE_L)
          SIG=SIG+F1
        END DO
      END IF
    END DO
  END DO

  CALL CURVATURE(T0,D%V(3)**T0)
  WRITE(*,*)"R0   ",R0
  IF (R0>MAG(QE-QS)/2) THEN
  NORV=G*UV(0.5D0*(QE-QS)+DSQRT((R0/MAG(QE-QS))**2-0.25)
+    *(D%V(3)**(QE-QS)))
  Q_E(I_P1,IL1,1)=QS
  DO
    NORV1=NORV
    I=1
    IF (PROP_DIPOLE(I_P1,IL1,1)/=0) THEN
        Q_T(I_P1,IL1,I)=D%V(3)**NORV
    END IF
    LL=0
    DO
      PLL=LL
      CALL CURVATURE(D%V(3)**NORV,G*NORV)
      Q_E(I_P1,IL1,I+1)=Q_E(I_P1,IL1,I)-R0*NORV+DT**(R0*NORV)
      NORV=DT**NORV
      IF (PROP_DIPOLE(I_P1,IL1,1)/=0) THEN
        Q_T(I_P1,IL1,I+1)=D%V(3)**NORV
      END IF
      I=I+1
      LL=MAG(Q_E(I_P1,IL1,I)-QS)
      IF ((MAG(Q_E(I_P1,IL1,I)-QS)>MAG(QE-QS)).OR.(LL<PLL)) EXIT
      !INSTABILITY CONDITION LL<PLL
    END DO
    DANG=UV(Q_E(I_P1,IL1,I)-QS)**UV(QE-QS)
    NORV=(H*DASIN(DANG%V(3)))**NORV1
    IF (MAG(DANG)<1.0D-2) EXIT
  END DO
  IF (LL<MAG(QE-QS)) THEN
    WRITE(*,*)"(2)LOOP IS UNSTABLE."
  END IF
  WRITE(11,FMT=16) "3DPOLY 0,0,0"
16 FORMAT(A6)
  AL=0
  N_Q(I_P1,IL1)=I
  Q_E(I_P1,IL1,I+1)=Q_E(I_P1,IL1,I)
  DO J=1,I
    QG=TRANS(ES(I_P))*Q_E(I_P1,IL1,J)+ORIGIN(I_P)
    WRITE(11,FMT=17) "  ",QG%V(1),",",QG%V(2),",",QG%V(3)
17   FORMAT(A2,E12.6,A1,E12.6,A1,E12.6)
    AL=AL+MAG(Q_E(I_P1,IL1,J+1)-Q_E(I_P1,IL1,J))
  END DO
  WRITE(*,*)"STRAIN= ",(AL-LL)/LL
  ELSE
    WRITE(*,*)"(1)LOOP IS UNSTABLE."
  END IF
  END SUBROUTINE SINGLE_EQUILIBRIUM
```

```
!***************************************************************!

     SUBROUTINE TAN_NOR(ILL)
USE VECTORS;USE VARIABLES
TYPE(VECTOR)::V21,V23,V1V2,V13,VV,P,Q
DOUBLE PRECISION::DD,S,ARC,TU,R0S,R0E,LS
DOUBLE PRECISION, DIMENSION(MAX_NODE)::UU
INTEGER::I,J;INTEGER,INTENT(IN)::ILL
IF (LOOPTYPE(I_P,ILL)==0) THEN
 PL(I_P,ILL,0)=PL(I_P,ILL,N_NODE(I_P,ILL))
 PL(I_P,ILL,N_NODE(I_P,ILL)+1)=PL(I_P,ILL,1)
 TL(I_P,ILL,0)=TL(I_P,ILL,N_NODE(I_P,ILL))
 TL(I_P,ILL,N_NODE(I_P,ILL)+1)=TL(I_P,ILL,1)
 NL(I_P,ILL,0)=NL(I_P,ILL,N_NODE(I_P,ILL))
 NL(I_P,ILL,N_NODE(I_P,ILL)+1)=NL(I_P,ILL,1)
ELSE
 TU=0
 DO I=1,N_NODE(I_P,ILL)-1
   TU=TU+MAG(PL(I_P,ILL,I+1)-PL(I_P,ILL,I))
   UU(I+1)=TU
 END DO
END IF
DO I=1+LOOPTYPE(I_P,ILL),N_NODE(I_P,ILL)-LOOPTYPE(I_P,ILL)
 V21=PL(I_P,ILL,I-1)-PL(I_P,ILL,I)
 V23=PL(I_P,ILL,I+1)-PL(I_P,ILL,I)
 P=V21**V23
 V1V2=H*(PL(I_P,ILL,I)+PL(I_P,ILL,I-1))
 V13=PL(I_P,ILL,I+1)-PL(I_P,ILL,I-1)
 IF (P%V(3)/=0) THEN
   S=(V13*V23)/(P%V(3))/2
 END IF
 VV=V1V2+S*(D%V(3)**V21)
 Q=PL(I_P,ILL,I)-VV
 DD=MAG(Q)
 IF (P%V(3)<0) THEN
   CUR(I_P,ILL,I)=1/DD
 ELSE
   CUR(I_P,ILL,I)=-1/DD
 END IF
 IF (CUSP(I_P,ILL,I)==0) THEN
   IF ((CUSP(I_P,ILL,I-1)/=0).OR.(CUSP(I_P,ILL,I+1)/=0)) THEN
     CUR(I_P,ILL,I)=(0.5+NR(I_P)*0.5)*CUR(I_P,ILL,I)
   END IF
 ELSE
   IF ((CUSP(I_P,ILL,I)==-2).OR.(CUSP(I_P,ILL,I)==2)) THEN
     CUR(I_P,ILL,I)=0
   ELSE
     CUR(I_P,ILL,I)=(0.3+NR(I_P)*0.7)*CUR(I_P,ILL,I)
   END IF
 END IF
 IF (I_TIME<2) THEN
   TL(I_P,ILL,I)=D%V(3)**Q/DD
 END IF !PLANE
 NL(I_P,ILL,I)=D%V(3)**TL(I_P,ILL,I)
 IF (LOOPTYPE(I_P,ILL)==1) THEN
   IF (I==2) THEN
     Q=PL(I_P,ILL,1)-VV
     DD=MAG(Q)
     TL(I_P,ILL,1)=D%V(3)**Q/DD
     NL(I_P,ILL,1)=G*Q/DD
     R0S=RR0(I_P,ILL)
     CUR(I_P,ILL,1)=1/R0S
     !CUR(I_P,ILL,1)=1/DD
   END IF !PLANE
   IF (I==N_NODE(I_P,ILL)-1) THEN
     Q=PL(I_P,ILL,N_NODE(I_P,ILL))-VV
```

```
      DD=MAG(Q)

      TL(I_P,ILL,N_NODE(I_P,ILL))=D%V(3)**Q/DD
      NL(I_P,ILL,N_NODE(I_P,ILL))=G*Q/DD
      R0E=RR0(I_P,ILL)
      CUR(I_P,ILL,N_NODE(I_P,ILL))=1/R0E
      !CUR(I_P,ILL,N_NODE(I_P,ILL))=1/DD

     END IF!PLANE
    END IF
   END DO
   LS=MAG(PL(I_P,ILL,N_NODE(I_P,ILL))-PL(I_P,ILL,1))
   DO I=1,N_NODE(I_P,ILL)-LOOPTYPE(I_P,ILL)
    IF (LOOPTYPE(I_P,ILL)==1) THEN
     IF (UU(I+1)/TU<0.5) THEN
      CUR(I_P,ILL,I+1)=1/((1-(2*UU(I+1)/TU)**0.5)*R0S+
+                    (2*UU(I+1)/TU)**0.5*(3*LS))
     ELSE
      CUR(I_P,ILL,I+1)=1/((1-(1-2*(UU(I+1)/TU-0.5))**0.5)*R0E+
+                   (1-2*(UU(I+1)/TU-0.5))**0.5*(3*LS))
     END IF
    END IF
    ARC=MAG(PL(I_P,ILL,I)-PL(I_P,ILL,I+1))!PLANE
   ! write(*,*)"ILL I CUR  ARC ",ILL,I,CUR(I_P,ILL,I),ARC
    IF (ABS(CUR(I_P,ILL,I))<1/ARC) THEN
     N2(I_P,ILL,I)=ABS(CUR(I_P,ILL,I))*ARC**2
     T2(I_P,ILL,I)=ARC
    ELSE
     N2(I_P,ILL,I)=ARC
     T2(I_P,ILL,I)=DSQRT(N2(I_P,ILL,I)/ABS(CUR(I_P,ILL,I)))
    END IF
    IF (ABS(CUR(I_P,ILL,I+1))<1/ARC) THEN
     T1(I_P,ILL,I+1)=ARC
     N1(I_P,ILL,I+1)=ABS(CUR(I_P,ILL,I+1))*ARC**2
    ELSE
     N1(I_P,ILL,I+1)=ARC
     T1(I_P,ILL,I+1)=DSQRT(N1(I_P,ILL,I+1)/ABS(CUR(I_P,ILL,I+1)))
    END IF

   END DO
   END SUBROUTINE TAN_NOR

   !************************************************************!

      SUBROUTINE TANG(ILL,I,UU)
   USE VECTORS;USE VARIABLES
   INTEGER,INTENT(IN)::ILL,I;DOUBLE PRECISION,INTENT(IN)::UU
   DOUBLE PRECISION::U4,U3,U2
   U2=UU*UU;U3=U2*UU;U4=U3*UU
   RT=30*(U4-2*U3+U2)*(PL(I_P,ILL,I+1)-PL(I_P,ILL,I))+
+   (-15*U4+32*U3-18*U2+1)*T2(I_P,ILL,I)*TL(I_P,ILL,I)+
+   (-15*U4+28*U3-12*U2)*T1(I_P,ILL,I+1)*TL(I_P,ILL,I+1)+
+   (-2.5*U4+6*U3-4.5*U2+UU)*N2(I_P,ILL,I)*NL(I_P,ILL,I)+
+   (2.5*U4-4*U3+1.5*U2)*N1(I_P,ILL,I+1)*NL(I_P,ILL,I+1)
   END SUBROUTINE TANG

   !************************************************************!

      subroutine tangent_vec
    USE VECTORS; USE VARIABLES
         double precision::c,e
         double precision, dimension(2)::a
         TYPE(VECTOR), dimension(2)::f
         INTEGER::I,MM

         PL(I_P,IL,0)=PL(I_P,IL,N_NODE(I_P,IL))
```

```
            PL(I_P,IL,N_NODE(I_P,IL)+1)=PL(I_P,IL,1)

     !Solve the tangent vectors at each data point
     c=4
     a(1)=(-c+sqrt(c*c-4))/2; a(2)=(-c-sqrt(c*c-4))/2

       DO MM=1,2
       e=1;f(MM)=ZERO%V(1)
          do i=0,N_NODE(I_P,IL)-1
         f(MM)=f(MM)+3*e*(PL(I_P,IL,N_NODE(I_P,IL)+1-i)-
    +        PL(I_P,IL,N_NODE(I_P,IL)-1-i))
         e=e*a(MM)
          end do
         f(MM)=f(MM)/(1-e)
       END DO

     !Calculate tangent vectors
       TL(I_P,IL,1)=(a(1)*f(1)-a(2)*f(2))/(a(1)-a(2))
       TL(I_P,IL,N_NODE(I_P,IL))=(f(1)-f(2))/(a(1)-a(2))


                        TL(I_P,IL,0)=TL(I_P,IL,N_NODE(I_P,IL))
                        TL(I_P,IL,N_NODE(I_P,IL)+1)=TL(I_P,IL,1)


                  DO ii=2,N_NODE(I_P,IL)-1
                    TL(I_P,IL,ii)=3.0D0*(PL(I_P,IL,ii)-PL(I_P,IL,ii-2))-
    +                  4.0D0*TL(I_P,IL,ii-1)-TL(I_P,IL,ii-2)
                  END DO


                        end subroutine tangent_vec

            !***************************************************************!

            !***************************************************************!

            SUBROUTINE SEEKER(SIG_FEM,NODE_NUMBER,LOOP_NODE,
    +NEWDIF,E_LENG,CAPSULE_NODE,SIG_CAPSULE,K,NUMBER_OF_NODES)

            USE VECTORS
            USE VARIABLES

            IMPLICIT NONE

            INTEGER, INTENT(IN) :: NODE_NUMBER
            INTEGER, INTENT(IN) :: E_LENG
            TYPE (MATRIX), INTENT(IN) :: SIG_FEM


            TYPE (VECTOR), INTENT(INOUT) :: LOOP_NODE
            INTEGER, INTENT(INOUT) ::NUMBER_OF_NODES,K
            TYPE (VECTOR), INTENT(OUT) :: NEWDIF
            TYPE (MATRIX), DIMENSION(8),INTENT(INOUT) :: SIG_CAPSULE
            !! ARRAY OF SIG_CAPSULES ASSOCIATED WITH CAPSULE_NODE NODES
            TYPE (VECTOR), DIMENSION(8),INTENT(INOUT) :: CAPSULE_NODE
            !!ARRAY OF CLOSEST FIELD_POINTS TO THE LOOP NODE

            TYPE (VECTOR), SAVE :: DIF1,DIF2,DIF3,DIF4,DIF5,DIF6,DIF7,DIF8

            INTEGER::GG
            !
            !VARIABLE DEFINITIONS:
            !CAPSULE_NODE(I) :THE ARRAY THAT CONTAINS THE FIELD NODES THAT SURROUND THE LOOP NODE
            !SIG_CAPSULE(I) :THE ARRAY THAT CONTAINS THE SIG_CAPSULEES AT EACH OF THE EIGHT FIELD
NODES
```

```
          !SIG_APP : THE APPLIED SIG_CAPSULE
          !
          !FOR THE FIRST DO LOOP, THE ARRAYS MUST BE INITIALIZED
          !


          IF (K == 1) THEN

           CAPSULE_NODE(1)= FEM_NODE
           CAPSULE_NODE(2)= FEM_NODE
           CAPSULE_NODE(3)= FEM_NODE
           CAPSULE_NODE(4)= FEM_NODE
           CAPSULE_NODE(5)= FEM_NODE
           CAPSULE_NODE(6)= FEM_NODE
           CAPSULE_NODE(7)= FEM_NODE
           CAPSULE_NODE(8)= FEM_NODE

          ENDIF

          !IF THE LOOP NODE FALLS WITHIN .001 UNITS OF THE FIELD NODE,
          !THE CAPSULE WILL BE POPULATED WITH FIELD NODE VALUES, AND
          !EACH CAPSULE NODE WILL TAKE ON THE FIELD NODE STRESS.



          IF ((ABS(LOOP_NODE%V(1)-FEM_NODE%V(1)) <= .00001).AND.
+   ( ABS(LOOP_NODE%V(2)-FEM_NODE%V(2)) <= .00001)  .AND.
+   ( ABS(LOOP_NODE%V(3)-FEM_NODE%V(3)) <= .00001)) THEN

      DO GG=1, 8

      CAPSULE_NODE(GG)=FEM_NODE
            SIG_CAPSULE(GG) = SIG_FEM

            ENDDO

    K = NUMBER_OF_NODES

    PRINT*,"SEEKER ACHIEVED A CLOSE MATCH."

          ELSE

          !!CHECK FOR MATCHING COORDINATES, GIVE LOOP NODE VALUES A NUDGE.

          IF ((ABS(LOOP_NODE%V(1)-FEM_NODE%V(1))<.000001).AND.
+   (ABS(LOOP_NODE%V(2)-FEM_NODE%V(2))<.000001).AND.
+   (LOOP_NODE%V(3)/=FEM_NODE%V(3)))THEN

 LOOP_NODE%V(1)=LOOP_NODE%V(1)+.000001
 LOOP_NODE%V(2)=LOOP_NODE%V(2)+.000001
 PRINT*, "NUDGED"

 ELSE IF((ABS(LOOP_NODE%V(1)-FEM_NODE%V(1))<.000001).AND.
+      (ABS(LOOP_NODE%V(3)-FEM_NODE%V(3))<.000001).AND.
+         (LOOP_NODE%V(2)/=FEM_NODE%V(2)))THEN

 LOOP_NODE%V(1)=LOOP_NODE%V(1)+.000001
 LOOP_NODE%V(3)=LOOP_NODE%V(3)+.000001
 PRINT*, "NUDGED"

          ELSE IF((ABS(LOOP_NODE%V(2)-FEM_NODE%V(2))<.000001).AND.
+      (ABS(LOOP_NODE%V(3)-FEM_NODE%V(3))<.000001).AND.
+         (LOOP_NODE%V(1)/=FEM_NODE%V(1)))THEN

 LOOP_NODE%V(2)=LOOP_NODE%V(2)+.000001
 LOOP_NODE%V(3)=LOOP_NODE%V(3)+.000001
 PRINT*, "NUDGED"

          ELSE IF((ABS(LOOP_NODE%V(1)-FEM_NODE%V(1))<.000001).AND.
+      (LOOP_NODE%V(2)/=FEM_NODE%V(2)).AND.
+      (LOOP_NODE%V(3)/=FEM_NODE%V(3)))THEN

 LOOP_NODE%V(1)=LOOP_NODE%V(1)+.000001
 PRINT*, "NUDGED"

          ELSE IF((ABS(LOOP_NODE%V(2)-FEM_NODE%V(2))<.000001).AND.
+      (LOOP_NODE%V(1)/=FEM_NODE%V(1)).AND.
+      (LOOP_NODE%V(3)/=FEM_NODE%V(3)))THEN

 LOOP_NODE%V(2)=LOOP_NODE%V(2)+.000001
 PRINT*, "NUDGED"

          ELSE IF((ABS(LOOP_NODE%V(3)-FEM_NODE%V(3))<.000001).AND.
+      (LOOP_NODE%V(2)/=FEM_NODE%V(2)).AND.
+      (LOOP_NODE%V(1)/=FEM_NODE%V(1)))THEN

 LOOP_NODE%V(3)=LOOP_NODE%V(3)+.000001
 PRINT*, "NUDGED"

          ENDIF


          !THE DIFFERENCE BETWEEN THE VECTORS OF INTEREST DEFINED

          NEWDIF = LOOP_NODE-FEM_NODE

          DIF1 = LOOP_NODE-CAPSULE_NODE(1)
          DIF2 = LOOP_NODE-CAPSULE_NODE(2)
          DIF3 = LOOP_NODE-CAPSULE_NODE(3)
          DIF4 = LOOP_NODE-CAPSULE_NODE(4)
          DIF5 = LOOP_NODE-CAPSULE_NODE(5)
          DIF6 = LOOP_NODE-CAPSULE_NODE(6)
          DIF7 = LOOP_NODE-CAPSULE_NODE(7)
          DIF8 = LOOP_NODE-CAPSULE_NODE(8)


          IF ((MAG(NEWDIF)<=MAG(DIF1)).AND.(ABS(NEWDIF%V(1))<E_LENG).AND.
+(ABS(NEWDIF%V(2))<E_LENG).AND.(ABS(NEWDIF%V(3))<E_LENG)) THEN



               CAPSULE_NODE(8) = CAPSULE_NODE(7)
               CAPSULE_NODE(7) = CAPSULE_NODE(6)
               CAPSULE_NODE(6) = CAPSULE_NODE(5)
               CAPSULE_NODE(5) = CAPSULE_NODE(4)
               CAPSULE_NODE(4) = CAPSULE_NODE(3)
               CAPSULE_NODE(3) = CAPSULE_NODE(2)
               CAPSULE_NODE(2) = CAPSULE_NODE(1)
               CAPSULE_NODE(1) = FEM_NODE

               SIG_CAPSULE(8) = SIG_CAPSULE(7)
               SIG_CAPSULE(7) = SIG_CAPSULE(6)
               SIG_CAPSULE(6) = SIG_CAPSULE(5)
               SIG_CAPSULE(5) = SIG_CAPSULE(4)
               SIG_CAPSULE(4) = SIG_CAPSULE(3)
               SIG_CAPSULE(3) = SIG_CAPSULE(2)
               SIG_CAPSULE(2) = SIG_CAPSULE(1)
               SIG_CAPSULE(1) = SIG_FEM


          ELSEIF ((MAG(NEWDIF)<=MAG(DIF2)).AND.(ABS(NEWDIF%V(1))<E_LENG)
+.AND.(ABS(NEWDIF%V(2))<E_LENG).AND.(ABS(NEWDIF%V(3))<E_LENG)) THEN
```

```
                CAPSULE_NODE(8) = CAPSULE_NODE(7)
                CAPSULE_NODE(7) = CAPSULE_NODE(6)
                CAPSULE_NODE(6) = CAPSULE_NODE(5)
                CAPSULE_NODE(5) = CAPSULE_NODE(4)
                CAPSULE_NODE(4) = CAPSULE_NODE(3)
                CAPSULE_NODE(3) = CAPSULE_NODE(2)
                CAPSULE_NODE(2) = FEM_NODE

                SIG_CAPSULE(8) = SIG_CAPSULE(7)
                SIG_CAPSULE(7) = SIG_CAPSULE(6)
                SIG_CAPSULE(6) = SIG_CAPSULE(5)
                SIG_CAPSULE(5) = SIG_CAPSULE(4)
                SIG_CAPSULE(4) = SIG_CAPSULE(3)
                SIG_CAPSULE(3) = SIG_CAPSULE(2)
                SIG_CAPSULE(2) = SIG_FEM


          ELSEIF ((MAG(NEWDIF)<=MAG(DIF3)).AND.(ABS(NEWDIF%V(1))<E_LENG)
     +.AND.(ABS(NEWDIF%V(2))<E_LENG).AND.(ABS(NEWDIF%V(3))<E_LENG)) THEN

                CAPSULE_NODE(8) = CAPSULE_NODE(7)
                CAPSULE_NODE(7) = CAPSULE_NODE(6)
                CAPSULE_NODE(6) = CAPSULE_NODE(5)
                CAPSULE_NODE(5) = CAPSULE_NODE(4)
                CAPSULE_NODE(4) = CAPSULE_NODE(3)
                CAPSULE_NODE(3) = FEM_NODE

                SIG_CAPSULE(8) = SIG_CAPSULE(7)
                SIG_CAPSULE(7) = SIG_CAPSULE(6)
                SIG_CAPSULE(6) = SIG_CAPSULE(5)
                SIG_CAPSULE(5) = SIG_CAPSULE(4)
                SIG_CAPSULE(4) = SIG_CAPSULE(3)
                SIG_CAPSULE(3) = SIG_FEM

          ELSEIF ((MAG(NEWDIF)<=MAG(DIF4)).AND.(ABS(NEWDIF%V(1))<E_LENG)
     +.AND.(ABS(NEWDIF%V(2))<E_LENG).AND.(ABS(NEWDIF%V(3))<E_LENG)) THEN

                CAPSULE_NODE(8) = CAPSULE_NODE(7)
                CAPSULE_NODE(7) = CAPSULE_NODE(6)
                CAPSULE_NODE(6) = CAPSULE_NODE(5)
                CAPSULE_NODE(5) = CAPSULE_NODE(4)
                CAPSULE_NODE(4) = FEM_NODE

                SIG_CAPSULE(8) = SIG_CAPSULE(7)
                SIG_CAPSULE(7) = SIG_CAPSULE(6)
                SIG_CAPSULE(6) = SIG_CAPSULE(5)
                SIG_CAPSULE(5) = SIG_CAPSULE(4)
                SIG_CAPSULE(4) = SIG_FEM


          ELSEIF ((MAG(NEWDIF)<=MAG(DIF5)).AND.(ABS(NEWDIF%V(1))<E_LENG)
     +.AND.(ABS(NEWDIF%V(2))<E_LENG).AND.(ABS(NEWDIF%V(3))<E_LENG)) THEN

                CAPSULE_NODE(8) = CAPSULE_NODE(7)
                CAPSULE_NODE(7) = CAPSULE_NODE(6)
                CAPSULE_NODE(6) = CAPSULE_NODE(5)
                CAPSULE_NODE(5) = FEM_NODE

                SIG_CAPSULE(8) = SIG_CAPSULE(7)
                SIG_CAPSULE(7) = SIG_CAPSULE(6)
                SIG_CAPSULE(6) = SIG_CAPSULE(5)
                SIG_CAPSULE(5) = SIG_FEM
```

```
        ELSEIF ((MAG(NEWDIF)<=MAG(DIF6)).AND.(ABS(NEWDIF%V(1))<E_LENG)
+.AND.(ABS(NEWDIF%V(2))<E_LENG).AND.(ABS(NEWDIF%V(3))<E_LENG)) THEN

            CAPSULE_NODE(8) = CAPSULE_NODE(7)
            CAPSULE_NODE(7) = CAPSULE_NODE(6)
            CAPSULE_NODE(6) = FEM_NODE

            SIG_CAPSULE(8) = SIG_CAPSULE(7)
            SIG_CAPSULE(7) = SIG_CAPSULE(6)
            SIG_CAPSULE(6) = SIG_FEM


        ELSEIF ((MAG(NEWDIF)<=MAG(DIF7)).AND.(ABS(NEWDIF%V(1))<E_LENG)
+.AND.(ABS(NEWDIF%V(2))<E_LENG).AND.(ABS(NEWDIF%V(3))<E_LENG)) THEN

            CAPSULE_NODE(8) = CAPSULE_NODE(7)
            CAPSULE_NODE(7) = FEM_NODE

            SIG_CAPSULE(8) = SIG_CAPSULE(7)
            SIG_CAPSULE(7) = SIG_FEM



        ELSEIF ((MAG(NEWDIF)<=MAG(DIF8)).AND.(ABS(NEWDIF%V(1))<E_LENG)
+.AND.(ABS(NEWDIF%V(2))<E_LENG).AND.(ABS(NEWDIF%V(3))<E_LENG)) THEN

            CAPSULE_NODE(8) = FEM_NODE

            SIG_CAPSULE(8) = SIG_FEM



 ENDIF

 ENDIF




 END SUBROUTINE SEEKER


        !************************************************************!
        !************************************************************!

        SUBROUTINE ARRANGER(CAPSULE_NODE,SIG_CAPSULE,E_LENG)

        USE VECTORS

        IMPLICIT NONE

        INTEGER :: H,CC
        TYPE (VECTOR), DIMENSION(8) :: SHAPE_POSITION
        TYPE (MATRIX), DIMENSION(8) :: SHAPE_FACTOR_STRESS
        TYPE (VECTOR) :: LOCAL_ORIGIN


        TYPE (VECTOR), DIMENSION(8), INTENT(INOUT) :: CAPSULE_NODE
        TYPE (MATRIX), DIMENSION(8), INTENT(INOUT) :: SIG_CAPSULE
        INTEGER, INTENT(IN) :: E_LENG
        !
        !VARIABLE DEFINITIONS:
```

```
        !CAPSULE_NODE(I) :THE ARRAY THAT CONTAINS THE FIELD NODES THAT
        !SURROUND THE LOOP NODE
        !SIG_CAPSULE(I) :THE ARRAY THAT CONTAINS THE STRESSES AT EACH OF
        !THE EIGHT FIELD NODES
        !SHAPE_POSITION(I) :: AN ARRAY CONTAINING THE ARRANGED VALUES FOR THE
        !FIELD NODES SURROUNDING THE LOOP NODE.  THESE VALUES ARE IN A
        !SPECIFIC ORDER FOR PROPER ASSIGNMENT TO THE SHAPE FUNCTIONS
        !SHAPE_FACTOR_STRESS(I) ::AN ARRAY CONTAINING THE STRESS ON THE
        !CAPSULE_NODE NODES IN THE APPROPRIATE POSITION WITH RESPECT
        !TO THE SHAPE FUNCTIONS
        !LOCAL_ORIGIN ::THE LOCAL ORIGIN OF THE CAPSULE_NODE FOR
        !ORIENTATION OF THE CAPSULE_NODE NODES ACCORDING
        !TO SHAPE FUNCTION REQUIREMENTS.  THIS ORIGIN IS NOT IN THE
        !CENTER OF THE CAPSULE_NODE.
        !

        !IF THE CAPSULE NODES ARE THE SAME, THEN SEEKER HAS DETERMINED THAT
        !THE LOOP NODE IS VERY CLOSE TO A FIELD NODE AND THUS THE CAPSULE
        !HAS TAKEN ON THESE FIELD NODE VALUES AND THE ASSOCIATED STRESS VALUES.


        IF (((CAPSULE_NODE(1)%V(1)) /= (CAPSULE_NODE(2)%V(1))).OR.
     +  ((CAPSULE_NODE(1)%V(2)) /= (CAPSULE_NODE(2)%V(2))).OR.
     +  ((CAPSULE_NODE(1)%V(3)) /= (CAPSULE_NODE(2)%V(3)))) THEN


        !ORIENT THE CAPSULE_NODE NODES.

        !FIND THE FIELD NODE ASSOCIATED WITH THE LOCAL ORIGIN OF THE
        !CAPSULE_NODE NODES


         IF (MAG(CAPSULE_NODE(1)) <= MAG(CAPSULE_NODE(2))) THEN
         LOCAL_ORIGIN = CAPSULE_NODE(1)

         ELSE
         LOCAL_ORIGIN = CAPSULE_NODE(2)
         ENDIF

         IF (MAG(CAPSULE_NODE(3)) <= MAG(LOCAL_ORIGIN)) THEN
         LOCAL_ORIGIN = CAPSULE_NODE(3)
         ENDIF

         IF (MAG(CAPSULE_NODE(4)) <= MAG(LOCAL_ORIGIN)) THEN
         LOCAL_ORIGIN = CAPSULE_NODE(4)
         ENDIF

         IF (MAG(CAPSULE_NODE(5)) <= MAG(LOCAL_ORIGIN)) THEN
         LOCAL_ORIGIN = CAPSULE_NODE(5)
         ENDIF

         IF (MAG(CAPSULE_NODE(6)) <= MAG(LOCAL_ORIGIN)) THEN
         LOCAL_ORIGIN = CAPSULE_NODE(6)
         ENDIF

         IF (MAG(CAPSULE_NODE(7)) <= MAG(LOCAL_ORIGIN)) THEN
         LOCAL_ORIGIN = CAPSULE_NODE(7)
         ENDIF

         IF (MAG(CAPSULE_NODE(8)) <= MAG(LOCAL_ORIGIN)) THEN
         LOCAL_ORIGIN = CAPSULE_NODE(8)
         ENDIF


        !COMPARE ALL OF THE CAPSULE_NODE NODES TO THE SEVEN DEFINED POSITIONS
        !IN THE CAPSULE BY ASSESSING THE CAPSULE NODES POSITION WITH RESPECT
        !TO THE LOCAL ORIGIN. ASSIGN A SPECIFIC POSITION VALUE TO THE NODES BASED ON
        !THIS CRITERIA. THEN REASSIGN THE CAPSULE_NODE AND SIG_CAPSULE VALUES.

         DO H=1,8

                IF ((CAPSULE_NODE(H)%V(1) == LOCAL_ORIGIN%V(1)).AND.
     +  (CAPSULE_NODE(H)%V(2) == LOCAL_ORIGIN%V(2)).AND.
     +  (CAPSULE_NODE(H)%V(3) == LOCAL_ORIGIN%V(3))) THEN

                SHAPE_POSITION(1) = CAPSULE_NODE(H)
                SHAPE_FACTOR_STRESS(1) = SIG_CAPSULE(H)

                ELSEIF ((CAPSULE_NODE(H)%V(1) == (LOCAL_ORIGIN%V(1)+E_LENG))
     +  .AND.(CAPSULE_NODE(H)%V(2) == LOCAL_ORIGIN%V(2)).AND.
     +  (CAPSULE_NODE(H)%V(3) == LOCAL_ORIGIN%V(3))) THEN

                SHAPE_POSITION(2) = CAPSULE_NODE(H)
                SHAPE_FACTOR_STRESS(2) = SIG_CAPSULE(H)

                ELSEIF ((CAPSULE_NODE(H)%V(1) == LOCAL_ORIGIN%V(1)).AND.
     +  (CAPSULE_NODE(H)%V(2) == (LOCAL_ORIGIN%V(2)+E_LENG)).AND.
     +  (CAPSULE_NODE(H)%V(3) == LOCAL_ORIGIN%V(3))) THEN

                SHAPE_POSITION(3) = CAPSULE_NODE(H)
                SHAPE_FACTOR_STRESS(3) = SIG_CAPSULE(H)

                ELSEIF ((CAPSULE_NODE(H)%V(1) == (LOCAL_ORIGIN%V(1)+E_LENG))
     +  .AND.(CAPSULE_NODE(H)%V(2) == (LOCAL_ORIGIN%V(2)+E_LENG))
     +  .AND.(CAPSULE_NODE(H)%V(3) == LOCAL_ORIGIN%V(3))) THEN

                SHAPE_POSITION(4) = CAPSULE_NODE(H)
                SHAPE_FACTOR_STRESS(4) = SIG_CAPSULE(H)

                ELSEIF ((CAPSULE_NODE(H)%V(1) == LOCAL_ORIGIN%V(1)).AND.
     +  (CAPSULE_NODE(H)%V(2) == LOCAL_ORIGIN%V(2)).AND.
     +  (CAPSULE_NODE(H)%V(3) == (LOCAL_ORIGIN%V(3)+E_LENG))) THEN

                SHAPE_POSITION(5) = CAPSULE_NODE(H)
                SHAPE_FACTOR_STRESS(5) = SIG_CAPSULE(H)

                ELSEIF ((CAPSULE_NODE(H)%V(1) == (LOCAL_ORIGIN%V(1)+E_LENG))
     +  .AND.(CAPSULE_NODE(H)%V(2) == LOCAL_ORIGIN%V(2)).AND.
     +  (CAPSULE_NODE(H)%V(3) == (LOCAL_ORIGIN%V(3)+E_LENG))) THEN

                SHAPE_POSITION(6) = CAPSULE_NODE(H)
                SHAPE_FACTOR_STRESS(6) = SIG_CAPSULE(H)

                ELSEIF ((CAPSULE_NODE(H)%V(1) == LOCAL_ORIGIN%V(1)).AND.
     +  (CAPSULE_NODE(H)%V(2) == (LOCAL_ORIGIN%V(2)+E_LENG)).AND.
     +  (CAPSULE_NODE(H)%V(3) == (LOCAL_ORIGIN%V(3)+E_LENG))) THEN

                SHAPE_POSITION(7) = CAPSULE_NODE(H)
                SHAPE_FACTOR_STRESS(7) = SIG_CAPSULE(H)

                ELSEIF ((CAPSULE_NODE(H)%V(1) == (LOCAL_ORIGIN%V(1)+E_LENG))
     +  .AND.(CAPSULE_NODE(H)%V(2) == (LOCAL_ORIGIN%V(2)+E_LENG))
     +  .AND.(CAPSULE_NODE(H)%V(3) == (LOCAL_ORIGIN%V(3)+E_LENG)))
     +  THEN

                SHAPE_POSITION(8) = CAPSULE_NODE(H)
                SHAPE_FACTOR_STRESS(8) = SIG_CAPSULE(H)

                else
                print*, "ARRANGER MISTAKE!!??"

                PRINT*, "LOCAL ORIGIN:",LOCAL_ORIGIN
                PRINT*, "ELEMENT LENGTH:", E_LENG
                DO CC=1,8
                PRINT*, "CAPSULE NODE AND MAGNITUDE:",
+       CAPSULE_NODE(CC), MAG(CAPSULE_NODE(CC))
                ENDDO
                PAUSE

                END IF

         END DO

          DO H=1,8

                CAPSULE_NODE(H) = SHAPE_POSITION(H)
                SIG_CAPSULE(H)  = SHAPE_FACTOR_STRESS(H)

          END DO

        END IF

        END SUBROUTINE ARRANGER

        !*************************************************************!
        !*************************************************************!

        SUBROUTINE LOOP_NODE_STRESS_ESTIMATOR
+               (CAPSULE_NODE,SIG_CAPSULE,E_LENG,LOOP_NODE)

        USE VECTORS
        USE VARIABLES
        IMPLICIT NONE

        INTEGER :: A,B,C
        TYPE(VECTOR) ::CENTRAL_CAPSULE_ORIGIN, LOCAL_LOOP_NODE
        DOUBLE PRECISION, DIMENSION(8) :: N


        INTEGER, INTENT(IN) ::E_LENG

        TYPE(MATRIX), DIMENSION(8), INTENT(IN) ::SIG_CAPSULE
        TYPE(VECTOR), DIMENSION(8), INTENT(IN) ::CAPSULE_NODE
        TYPE(VECTOR), INTENT (IN)::LOOP_NODE


        !DEFINE VARIABLES:
        !LOCAL_LOOP_NODE :: THE LOCAL LOOP NODE VALUE
        !CENTRAL_CAPSULE_ORIGIN :: THE CENTER OF THE CAPSULE IN GLOBAL COORDINATES
        !CAPSULE_NODE(1) :: ARRANGED SUCH THAT IT IS THE ORIGIN OF THE
        !CAPSULE(AT A CORNER)
        !P::LOOP NODE


        !CALCULATE THE THREE HALF DIMENSIONS OF THE CAPSULE A, B, C

        A = E_LENG/2
        B = E_LENG/2
        C = E_LENG/2

        !CALCULATE THE POSITION OF THE LOOP NODE WITH RESPECT TO THE CAPSULE
        !BY SUBTRACTING THE LOOP NODE VECTOR FROM THE CENTRAL ORIGIN OF THE
        !CAPSULE (IN GLOBAL COORDINATES).

        CENTRAL_CAPSULE_ORIGIN%V(1) = CAPSULE_NODE(1)%V(1) + A
        CENTRAL_CAPSULE_ORIGIN%V(2) = CAPSULE_NODE(1)%V(2) + B
        CENTRAL_CAPSULE_ORIGIN%V(3) = CAPSULE_NODE(1)%V(3) + C
```

```
          LOCAL_LOOP_NODE = LOOP_NODE - CENTRAL_CAPSULE_ORIGIN


          !CALCULATE THE SHAPE FUNCTIONS AND THE STRESS AT THE LOOP NODE
          !X, Y, AND Z VALUES ARE THE VALUES
          !OF THE LOOP NODE COMPONENTS - THE CENTRAL_CAPSULE_ORIGIN.
          !

          N(1)=(((A-LOCAL_LOOP_NODE%V(1))*(B-LOCAL_LOOP_NODE%V(2))*
+(C-LOCAL_LOOP_NODE%V(3))))/(8*A*B*C))

          N(2)=(((A+LOCAL_LOOP_NODE%V(1))*(B-LOCAL_LOOP_NODE%V(2))*
+(C-LOCAL_LOOP_NODE%V(3))))/(8*A*B*C))

          N(3)=(((A-LOCAL_LOOP_NODE%V(1))*(B+LOCAL_LOOP_NODE%V(2))*
+(C-LOCAL_LOOP_NODE%V(3))))/(8*A*B*C))

          N(4)=(((A+LOCAL_LOOP_NODE%V(1))*(B+LOCAL_LOOP_NODE%V(2))*
+(C-LOCAL_LOOP_NODE%V(3))))/(8*A*B*C))

          N(5)=(((A-LOCAL_LOOP_NODE%V(1))*(B-LOCAL_LOOP_NODE%V(2))*
+(C+LOCAL_LOOP_NODE%V(3))))/(8*A*B*C))

          N(6)=(((A+LOCAL_LOOP_NODE%V(1))*(B-LOCAL_LOOP_NODE%V(2))*
+(C+LOCAL_LOOP_NODE%V(3))))/(8*A*B*C))

          N(7)=(((A-LOCAL_LOOP_NODE%V(1))*(B+LOCAL_LOOP_NODE%V(2))*
+(C+LOCAL_LOOP_NODE%V(3))))/(8*A*B*C))

          N(8)=(((A+LOCAL_LOOP_NODE%V(1))*(B+LOCAL_LOOP_NODE%V(2))*
+(C+LOCAL_LOOP_NODE%V(3))))/(8*A*B*C))


          SIG_ESTIMATED=((N(1)*SIG_CAPSULE(1))+(N(2)*SIG_CAPSULE(2))+
+(N(3)*SIG_CAPSULE(3))+(N(4)*SIG_CAPSULE(4))+
+(N(5)*SIG_CAPSULE(5))+(N(6)*SIG_CAPSULE(6))+
+(N(7)*SIG_CAPSULE(7))+(N(8)*SIG_CAPSULE(8)))/MU

          !!SIG_LOOP_NODE IS DIMENSIONLESS


          END SUBROUTINE LOOP_NODE_STRESS_ESTIMATOR


          !************************************************************!

      SUBROUTINE DEFORMATION

 USE VECTORS
 USE VARIABLES

 IMPLICIT NONE


 DOUBLE PRECISION,EXTERNAL::FEX,JEX
 DOUBLE PRECISION,ALLOCATABLE,DIMENSION(:)::ATOL,IWORK,RWORK,Y
 DOUBLE PRECISION::X1,Y1,X0,Y0,R,U0
 INTEGER::N,I,J,NN,NB,IT
 DOUBLE PRECISION:: RPAR,T, TOUT !,ATOLL,RTOL
 INTEGER::NEQ,ITOL,ITASK,ISTATE,IOPT,LRW,LIW,MF,IOUT,IPAR,NTIME
 DOUBLE PRECISION,DIMENSION(4,3)::SHAP
 TYPE(VECTOR)::RR


          !NEQ=NUMBER OF EQUATIONS
          !NB=RELATED TO NEQ
          !ATOL=
          !RTOL=
          !ATOLL=
          !N=TOTAL NUMBER OF NODES
          !I_P=PLANE NUMBER
          !IL=LOOP NUMBER
          !ELEMENT=SEGMENT BETWEEN TWO NODES
          !PL=LOCAL NODE POSITION IN THE SLIP PLANE
          !RR=?
          !Y=?


    NEQ=N_NODE(I_P,IL)*4
    NB=N_NODE(I_P,IL)*4

    ALLOCATE(ATOL(NEQ*4),Y(NEQ*4),RWORK(22 + 9*NEQ + 2*NEQ*NEQ)
   +       ,IWORK(30+NEQ))

    !ALLOCATE(ELEMENT(N_NODE(I_P,IL),2))


    N=N_NODE(I_P,IL)
    RTOL=1.0D-3
    ATOLL=1.0D-4

    T = 0.0D0
    TOUT = 0
    ITOL = 2
    DO I=1, NEQ;ATOL(I) = ATOLL;ENDDO


    ITASK = 1
    ISTATE = 1
    IOPT = 0
    LRW = 22 + 9*NEQ + 2*NEQ*NEQ
    LIW = 30+NEQ
    MF = 22

    DO I=1,N


      Y(4*I-3)=PL(I_P,IL,I)%V(1)
      Y(4*I-2)=PL(I_P,IL,I)%V(2)
      Y(4*I-1)=TL(I_P,IL,I)%V(1)
      Y(4*I  )=TL(I_P,IL,I)%V(2)


    ENDDO

    DO I=1,N
      ELEMENT(I,1)=I
      ELEMENT(I,2)=I+1


    ENDDO
    ELEMENT(N,2)=1





      TOUT = TOUT+DTIME
```

```
        !CALL DVODE(FEX,NEQ,Y,T,TOUT,ITOL,RTOL,ATOL,ITASK,ISTATE,

      !           IOPT,RWORK,LRW,IWORK,LIW,JEX,MF,RPAR,IPAR)


!  20     FORMAT(1X,es12.4,3es14.6)
        IF (ISTATE .LT. 0) THEN
          WRITE(6,90)ISTATE
          STOP
        ENDIF



        DO I=1,N
          PL(I_P,IL,I)%V(1)=Y(I*4-3)
          PL(I_P,IL,I)%V(2)=Y(I*4-2)
          TL(I_P,IL,I)%V(1)=Y(I*4-1)
          TL(I_P,IL,I)%V(2)=Y(I*4)

        ENDDO

 90   FORMAT(///' Error halt.. ISTATE =',I3)



      END SUBROUTINE DEFORMATION

      !*************************************************************!


      SUBROUTINE FEX (NEQ, T, Y, YDOT, RPAR, IPAR)
       USE VECTORS
       USE VARIABLES
       IMPLICIT NONE
       DOUBLE PRECISION, DIMENSION(NEQ,NEQ)::KG
       DOUBLE PRECISION, DIMENSION(NEQ)::FG
       DOUBLE PRECISION:: RPAR, T
       INTEGER::IPAR,NEQ,N,NB,I,J
       DOUBLE PRECISION,DIMENSION(NEQ)::Y,YDOT
       N=NEQ/4
        NB=NEQ
        DO I=1,N
          PL(I_P,IL,I)%V(1)=Y(4*I-3)
          PL(I_P,IL,I)%V(2)=Y(4*I-2)
          PL(I_P,IL,I)%V(3)=0.0D0
          TL(I_P,IL,I)%V(1)=Y(4*I-1)
          TL(I_P,IL,I)%V(2)=Y(4*I)
          TL(I_P,IL,I)%V(3)=0.0D0

        ENDDO

       CALL STIFFNESS(KG,NEQ,NB)
       CALL FORCE(YDOT,NEQ)
       CALL GAUSS(KG,YDOT,NEQ,NB)
      END SUBROUTINE FEX


!=====================================================


      SUBROUTINE FORCE(FG,NN)

      USE VECTORS
      USE VARIABLES
      IMPLICIT NONE
      INTEGER:: NN,NB
      DOUBLE PRECISION, DIMENSION(NN)::FG
      DOUBLE PRECISION, DIMENSION(8)::FE
      DOUBLE PRECISION, DIMENSION(20)::POS,WT
      TYPE(VECTOR)::R,RU,RUU,RR,P1,TT1,P2,TT2
      DOUBLE PRECISION,DIMENSION(4,3)::SHAP
      DOUBLE PRECISION::UU,DS,X,Y,KAPPA,FX,FY
      INTEGER::I,J,IE,II
      TYPE(VECTOR)::T_Q,
     +  APPLIED_F,SELF_F,PK_F,PEIRELS_F,FEM_F,TOTAL_F,
     +  RESULTANT_F_L !SIG_ESTIMATED

      TYPE(MATRIX)::TRANS_MATRIX  !SIG_Q,

       SIG_Q=ZERO
       SIG_ESTIMATED=ZERO


      CALL QUADRATURE(MAX_QUAD,POS,WT)
      DO I=1,NN;FG(I)=0.0;ENDDO
      DO IE=1,NN/4-1
       P1=PL(I_P,IL,ELEMENT(IE,1));TT1=TL(I_P,IL,ELEMENT(IE,1))
       P2=PL(I_P,IL,ELEMENT(IE,2));TT2=TL(I_P,IL,ELEMENT(IE,2))

       DO I=1,8;FE(I)=0.0;ENDDO
       DO I=1,MAX_QUAD
        U=(POS(I)+1)/2.0D0
        CALL GETSHAPE(SHAP,U)
        R=SHAP(1,1)*P1+SHAP(2, 1)*TT1+SHAP(3,1)*P2+SHAP(4,1)*TT2
        RU=SHAP(1,2)*P1+SHAP(2,2)*TT1+SHAP(3,2)*P2+SHAP(4,2)*TT2
        RUU=SHAP(1,3)*P1+SHAP(2,3)*TT1+SHAP(3,3)*P2+SHAP(4,3)*TT2
        DS=DSQRT(RU*RU)
        RR=RU**RUU
        KAPPA=DSQRT(RR*RR)/(DSQRT(RU*RU)) **3
        IF(RR%V(3) .LT. 0) THEN
           KAPPA=-KAPPA
        ENDIF

!  transform tangent vector from local to global coodinate
!       and make it unit vector
        !NEED TO CONVERT VECTOR VALUES TO LOCAL COORDINATES

        TRANS_MATRIX%V(3)=UV(MILLER_Q)
        TRANS_MATRIX%V(1)=UV(d%v(3)**MILLER_Q)
        TRANS_MATRIX%V(2)=TRANS_MATRIX%V(3)**TRANS_MATRIX%V(1)
        T_Q=TRANS(TRANS_MATRIX)*RU
        T_Q=UV(T_Q)
        CALL COMPUTE_FORCE(T_Q,KAPPA,SIG_Q,B_Q,MILLER_Q,
     +          RESULTANT_F_L,SIG_ESTIMATED)


        DO II=1,4
         FE(II*2-1)=FE(II*2-1)
     +             +WT(I)*0.5*DS*SHAP(II,1)*(-RESULTANT_F_L%V(1))
         FE(II*2)=FE(II*2)
     +             +WT(I)*0.5*DS*SHAP(II,1)*(-RESULTANT_F_L%V(2))
        ENDDO
       ENDDO

       DO I=1,4
        FG(ELEMENT(IE,1)*4-4+I)=FG(ELEMENT(IE,1)*4-4+I)+FE(I)
       ENDDO
       DO I=5,8
        FG(ELEMENT(IE,2)*4-8+I)=FG(ELEMENT(IE,2)*4-8+I)+FE(I)
       ENDDO
      ENDDO


      END SUBROUTINE FORCE



!==========================================================

      SUBROUTINE STIFFNESS(KG,NN,NB)
      USE VECTORS; USE VARIABLES
      IMPLICIT NONE
      INTEGER:: NN,NB
      TYPE(VECTOR):: P1,TT1,P2,TT2
      DOUBLE PRECISION,DIMENSION(NN,NB)::KG
      DOUBLE PRECISION,DIMENSION(8,8)::KE
      INTEGER ::I,DEG,J,K,L,I1,J1,N

      N=NN/4     !NUMBER OF SEGMENT

      DO I=1,NN;DO J=1,NB
         KG(I,J)=0.0D0
      END DO; ENDDO

      DO I=1, N-1
        P1=PL(I_P,IL,ELEMENT(I,1));TT1=TL(I_P,IL,ELEMENT(I,1))
        P2=PL(I_P,IL,ELEMENT(I,2));TT2=TL(I_P,IL,ELEMENT(I,2))

         CALL ELEMENTSTIFF(KE,P1,TT1,P2,TT2)

         DO K=1,4
           I1=ELEMENT(I,1)*4-4+K
           DO L=K,4
             J1=ELEMENT(I,1)*4-4+L
             KG(I1,J1-I1+1)=KG(I1,J1-I1+1)+KE(K,L)
           ENDDO
         ENDDO

         IF(ELEMENT(I,1) .LE. ELEMENT(I,2)) THEN
           DO K=1,4
            I1=ELEMENT(I,1)*4-4+K
            DO L=5,8
             J1=ELEMENT(I,2)*4-8+L
             KG(I1,J1-I1+1)=KG(I1,J1-I1+1)+KE(K,L)
            ENDDO
           ENDDO
         ELSE
            DO K=5,8
              I1=ELEMENT(I,2)*4-8+K
              DO L=1,4
                J1=ELEMENT(I,1)*4-4+L
                KG(I1,J1-I1+1)=KG(I1,J1-I1+1)+KE(K,L)
              ENDDO
            ENDDO
         ENDIF

         DO K=5,8
           I1=ELEMENT(I,2)*4-8+K
           DO L=K,8
             J1=ELEMENT(I,2)*4-8+L
             KG(I1,J1-I1+1)=KG(I1,J1-I1+1)+KE(K,L)
           ENDDO
         ENDDO

      ENDDO
      END SUBROUTINE STIFFNESS


!===========================================================

      SUBROUTINE ELEMENTSTIFF(KE,P1,TT1,P2,TT2)
      USE VECTORS; USE VARIABLES
      IMPLICIT NONE
      TYPE(VECTOR):: P1,P2,TT1,TT2,R,RU
      DOUBLE PRECISION, DIMENSION(20)::POS,WT
      DOUBLE PRECISION, DIMENSION(8,8)::KE
      DOUBLE PRECISION, DIMENSION(8,3)::SHAP
      INTEGER :: I,J,II,JJ
      DOUBLE PRECISION :: DS





      CALL QUADRATURE(MAX_QUAD,POS,WT)

      DO I=1,8;DO J=1,8
       KE(I,J)=0.0D0
      ENDDO; ENDDO

      DO I=1,MAX_QUAD
       U=(POS(I)+1)/2.0D0
       CALL GETSHAPE(SHAP,U)
       R=SHAP(1,1)*P1+SHAP(2,1)*TT1+SHAP(3,1)*P2+SHAP(4,1)*TT2
       RU=SHAP(1,2)*P1+SHAP(2,2)*TT1+SHAP(3,2)*P2+SHAP(4,2)*TT2
       DS=DSQRT(RU*RU)
       DO II=1,4      !  1,4
       DO JJ=1,4
        KE(II*2-1,JJ*2-1)=KE(II*2-1,JJ*2-1)
     +          +WT(I)*0.5*MOBILITY*SHAP(II,1)*SHAP(JJ,1)*DS
        KE(II*2,JJ*2)=KE(II*2-1,JJ*2-1)
       ENDDO
       ENDDO
      ENDDO

      END SUBROUTINE ELEMENTSTIFF

!=============================================================
      SUBROUTINE GETSHAPE(SHAP,UU)
      IMPLICIT NONE
      INTEGER::SEG_TYPE
      DOUBLE PRECISION, DIMENSION(4,3)::SHAP
      DOUBLE PRECISION::UU

        SHAP(1,1) = 2*UU**3-3*UU*UU+1
        SHAP(2,1) = UU**3-2*UU*UU+UU
        SHAP(3,1) =-2*UU**3+3*UU*UU
        SHAP(4,1) = UU**3-UU*UU

        SHAP(1,2)= 6*UU*UU-6*UU
        SHAP(2,2)=3*UU*UU-4*UU+1
        SHAP(3,2)=-6*UU*UU+6*UU
        SHAP(4,2)=3*UU*UU-2*UU

        SHAP(1,3)= 12*UU-6
        SHAP(2,3)=6*UU-4
        SHAP(3,3)=-12*UU+6
        SHAP(4,3)=6*UU-2
      END SUBROUTINE GETSHAPE


!=============================================================

      SUBROUTINE BOUNDARY(A,B,NT,ND,K)
      IMPLICIT NONE
            INTEGER ::NT,ND
      DOUBLE PRECISION,DIMENSION(NT,ND) :: A
      DOUBLE PRECISION,DIMENSION(NT):: B
      INTEGER ::K,I

      A(K,1)=1.0
      B(K)=0.0
      DO I=2, ND
        A(K,I)=0.0
      ENDDO

      DO I=K-1,K-ND+1,-1
        IF (I .GE. 0) A(I,K-I+1)=0.0
      ENDDO

      END SUBROUTINE BOUNDARY

!==============================================================

      SUBROUTINE GAUSS(A,B,NT,ND)
      IMPLICIT NONE
      INTEGER NT,ND
      DOUBLE PRECISION, DIMENSION(NT,ND) :: A
      DOUBLE PRECISION, DIMENSION(NT) :: B
      INTEGER :: K,L,J,M1,M2
      DOUBLE PRECISION:: C
        DO K=1,NT-1
          M1=NT-K
          IF (M1 .GT. ND-1) M1=ND-1
          DO L=1,M1
            C=A(K,L+1)/A(K,1)
            M2=ND-L
            DO J=1,M2
              A(K+L,J)=A(K+L,J)-C*A(K,J+L)
            END DO
            B(K+L)=B(K+L)-C*B(K)
          ENDDO
        ENDDO
        IF(A(NT,1)+1.0D0 .EQ. 1.0D0) THEN
           PRINT *, " FAIL TO SOLVE TO LINEAR EQUATIONS"
           STOP
        ENDIF
        B(NT)=B(NT)/A(NT,1)
        DO K=NT-1,1,-1
          C=B(K)
          M1=ND
          IF (NT-K+1 .LE. ND) M1=NT-K+1
          DO J=2,M1
                C=C-A(K,J)*B(K+J-1)
          ENDDO
          IF(A(K,1)+1.0D0 .EQ. 1.0D0) THEN
            PRINT *, " FAIL TO SOLVE TO LINEAR EQUATIONS"
            STOP
          ENDIF
          B(K)=C/A(K,1)
        ENDDO
      END SUBROUTINE GAUSS
```

```
    !*************************************************************!

    SUBROUTINE JEX (NEQ, T, Y, ML, MU, PD, NRPD, RPAR, IPAR)
    DOUBLE PRECISION PD, RPAR, T, Y
    DIMENSION Y(NEQ), PD(NRPD,NEQ)
!    PD(1,1) = -.04D0
!    PD(1,2) = 1.D4*Y(3)
!    PD(1,3) = 1.D4*Y(2)
!    PD(2,1) = .04D0
!    PD(2,3) = -PD(1,3)
!    PD(3,2) = 6.D7*Y(2)
!    PD(2,2) = -PD(1,2) - PD(3,2)
    RETURN
    END

********************************
```

APPENDIX B: Input And Output Files

Input Files:

UNIT=1, MATERIAL_INPUT.TXT	Properties, Analysis Type
UNIT=2, GEOMETRY_INPUT.TXT	Input/Output Geometry File
UNIT=3, INTERACTION_GEOM_INPUT.TXT	Dislocation Geometry
UNIT=4, FIELD_INPUT.TXT	Field Node Positions
UNIT=5, FEM_INPUT.TXT	Unused
UNIT=6, OBSTACLE_COOR_INPUT.TXT	Unused
UNIT=7, COORD_AND_FEM_STRESS_INPUT.TXT	Image Stress Field
UNIT=8, DYNAMIC_LOOP_DATA.TXT	Dyn. Loop Node Positions

Output Files:

UNIT=11, AUTOCAD_OUTPUT.SCR	AUTOCAD Information
UNIT=12, DISL_GEOM_OUTPUT.TXT	(unused)
UNIT=13, INTERACTION_OUTPUT.TXT	Force Distributions
UNIT=14, FEM_OUTPUT.TXT	Reversed Tractions
UNIT=15, ELASTIC_FIELD_OUTPUT.TXT	Surface Tractions
UNIT=16, LOOP_UPDATE.TXT	(unused)
UNIT=20, RESULTANT_F.TXT	Resultant Forces
UNIT=21, IMAGE_F.TXT	Image Forces
UNIT=22, PEIRELS_F.TXT	Peierls Forces
UNIT=23, APPLIED_F.TXT	Applied Forces
UNIT=24, SELF_F.TXT	Self Forces

APPENDIX C: Peripheral Programs

***Note:** The following programs were written for the purpose of linking ANSYS FEM output files with Microplasticity. Three FORTRAN90 programs are included below: CONVERT, COORDINATES_AND_NORMALS, and CORRDINATES_AND_STRESS. Each program begins with a comment describing its purpose.

```
PROGRAM CONVERT

!This program converts the output stress values from ANSYS to a continuous
!table of stress values.
!The first line read is a blank, hense the need for A.
IMPLICIT NONE
REAL SX, SY, SZ, SXY, SYZ, SXZ
INTEGER ::NODE,i
CHARACTER (80):: A
CHARACTER (8)::B
CHARACTER (14)::C
OPEN (UNIT = 1, FILE = 'ANSYS_STRESS.TXT')
OPEN (UNIT = 2, FILE = 'STRESS.TXT')
!FIND APPROPRIATE DATA TO BE READ FROM ANSYS_STRESS.TXT
WRITE (2,*) "  NODE# SX        SY        SZ        SXY       SYZ       SXZ"
READ (1,100)B
BACKSPACE (UNIT= 1)
READ (1,110)C
DO WHILE (C /= 'MINIMUM VALUES')
        READ (1,10) B
        DO WHILE (B .NE.  '    NODE')
                READ (1,10)B
        ENDDO
        A='INITIAL'
        DO WHILE (A /= '          ')
                READ (1,100)A
                IF (A /= '          ')THEN
                WRITE (2,100)A
                END IF
        ENDDO
        READ (1,110) C
ENDDO
100 FORMAT (BN,A81)
10 FORMAT (BN,A8)
110 FORMAT (1X,A14)
WRITE*,"STOP"
END PROGRAM CONVERT



PROGRAM COORDINATES_AND_NORMALS
!This program takes inputed values for the node numbers and coordinates from an
!ANSYS output file called COORDINATES.TXT and user input as to side length and
!number of divisions per side (from the FEM model),
!and outputs the inputed coordinates divided by the lattice constant (2.8 x 10^-10 m)
!along with a label indicating interior vs. surface and
!the normal to the surface for the particular node. The output coordinates are unitless.
IMPLICIT NONE
REAL :: XX, YY, ZZ, A, B, SIDE, AREA, NODES_PER_SIDE, NODAL_AREA , LATTICE
INTEGER :: NODE, DIV, NUMBER_OF_NODES
CHARACTER (1) :: COND
CHARACTER (5) :: SKIPPER
CHARACTER (8) :: ni,ti,bi
```

```
ti=" 0  0  0"
bi=" 0  0  0"
!NODE is the node number, DIV is the number of divisions per side from the
!meshing process, SIDE is the length of the side in meters, XX, YY and ZZ are the
!cartesian coordinates, A is half the length of an element in the mesh, B
!is the length of one side minus half an element length, ni is the normal vector itself,
!SKIPPER helps to eliminate non-data lines, and COND simply state the condition of the node
!(either on the surface or in the interior).
OPEN (UNIT = 1, FILE = 'COORDINATES.TXT')
OPEN (UNIT = 2, FILE = 'FIELD_POINTS.TXT')
PRINT*,'Please type in the dimension of one side of the FEM model in meters'
PRINT*, '(example: .000003).'
READ*, SIDE
PRINT*,"Please type in the lattice constant in m."
READ*,LATTICE
PRINT*, 'Please type in the number of divisions per side used in the meshing '
PRINT*, 'process (example: 10).'
READ*,DIV
READ(1,*)
READ(1,*)
READ(1,*)
READ(1,*)
READ(1,*)
NUMBER_OF_NODES = (DIV + 1) * (DIV + 1) * (DIV + 1)
AREA = SIDE * SIDE
NODES_PER_SIDE = (DIV + 1) * (DIV + 1)
NODAL_AREA = AREA / NODES_PER_SIDE
NODE = 1
WRITE (2,*)NUMBER_OF_NODES
WRITE (2,*)NODAL_AREA
DO WHILE (NODE <= (NUMBER_OF_NODES - 1))
        READ(1,300)SKIPPER
        300 FORMAT (3X,A)
        !SKIPPER causes read to skip over interruptions in data tables.
        IF (SKIPPER == "     ") THEN
        READ(1,*)
        READ(1,100)NODE, XX, YY, ZZ
        100 FORMAT (3X,I5,3X,F19.11,1X, F19.11,1X,F19.11)
        ELSEIF (SKIPPER == "NODE ") THEN
        READ(1,100)NODE, XX, YY, ZZ
        ELSE
        BACKSPACE (UNIT = 1)
        READ(1,100)NODE, XX, YY, ZZ
        ENDIF
        COND = "I"
        ni="********"
        A=SIDE/(2*DIV)
        B=SIDE-(SIDE/(2*DIV))
        !X FIELD NORMAL VALUES DEFINED
        IF (XX >B) THEN
        COND = "S"
        IF ((YY < B) .AND. (YY > A) .AND. (ZZ < B) .AND. (ZZ > A))THEN
        ni= "1  0  0"
        ENDIF
        ELSEIF (XX < A) THEN
        COND="S"
        IF ((YY < B) .AND. (YY > A) .AND. (ZZ < B) .AND. (ZZ > A))THEN
        ni= "-1  0  0"
        ENDIF
        ENDIF
        !Y FIELD NORMAL VALUES DEFINED
        IF (YY > B) THEN
        COND = "S"
        IF ((XX < B) .AND. (XX >A) .AND. (ZZ < B) .AND. (ZZ > A))THEN
        ni= " 0  1  0"
        ENDIF
        ELSEIF (YY < A) THEN
        COND="S"
        IF ((XX < B) .AND. (XX > A) .AND. (ZZ < B) .AND. (ZZ > A))THEN
        ni= " 0 -1  0"
        ENDIF
        ENDIF
        !Z FIELD NORMAL VALUES DEFINED
        IF (ZZ >B) THEN
        COND = "S"
        IF ((YY < B) .AND. (YY > A) .AND. (XX < B) .AND. (XX > A))THEN
        ni= " 0  0  1"
        ENDIF
        ELSEIF (ZZ < A) THEN
        COND="S"
        IF ((YY < B) .AND. (YY > A) .AND. (XX < B) .AND. (XX > A))THEN
        ni= " 0  0 -1"
        ENDIF
        ENDIF
        !X, Y PLANE INTERFACE NORMALS DEFINED
        IF ((XX >B) .AND. (YY > B)) THEN
        ni= " 0  1  0"
        ENDIF
        IF ((XX > B) .AND. (YY <A)) THEN
        ni= " 0 -1  0"
        ENDIF
        IF ((XX < A) .AND. (YY >B)) THEN
        ni= " 0  1  0"
        ENDIF
        IF ((XX <A) .AND. (YY < A)) THEN
        ni= " 0 -1  0"
        ENDIF
        !X, Z PLANE INTERFACE NORMALS DEFINED
        IF ((XX > B) .AND. (ZZ > B) .AND. (YY < B) .AND. (YY > A)) THEN
        ni= " 1  0  0"
        ENDIF
        IF ((XX > B) .AND. (ZZ < A) .AND. (YY < B) .AND. (YY >A)) THEN
        ni= " 1  0  0"
        ENDIF
        IF ((XX < A) .AND. (ZZ > B) .AND. (YY < B) .AND. (YY > A)) THEN
        ni= "-1  0  0"
        ENDIF
        IF ((XX < A) .AND. (ZZ < A) .AND. (YY < B) .AND. (YY > A)) THEN
        ni= "-1  0  0"
        ENDIF
        !Z, Y PLANE INTERFACE NORMALS DEFINED
        IF ((YY > B) .AND. (ZZ > B) .AND. (XX < B) .AND. (XX > A)) THEN
        ni= " 0  0  1"
        ENDIF
        IF ((YY < A) .AND. (ZZ < A) .AND. (XX < B) .AND. (XX >A)) THEN
        ni= " 0  0 -1"
        ENDIF
        IF ((YY < A) .AND. (ZZ > B) .AND. (XX < B) .AND. (XX > A)) THEN
        ni= " 0  0  1"
        ENDIF
        IF ((YY > B) .AND. (ZZ < A) .AND. (XX < B) .AND. (XX > A)) THEN
        ni= " 0  0 -1"
        ENDIF
        WRITE (2, 200)NODE, XX/(LATTICE), YY/(LATTICE), ZZ/(LATTICE), ni, ti, bi
        200 FORMAT (I5,3X,E8.2,3X,E8.2,3X,E8.2,3X, A,3X,A,3X,A)
ENDDO

PRINT*,"HASTA LA VISTA BABY!!."

END PROGRAM COORDINATES_AND_NORMALS
```

```
    !==================================================================!

    PROGRAM COORDINATES_AND_STRESS

    !This program takes the values in FIELD_PONITS.TXT and STRESS.TXT and
    !combines them in one file for use by MICROPLASTICITY.EXE in calculating
    !the Peach Kohler Force.
    USE VECTORS

    IMPLICIT NONE

    TYPE (VECTOR) ::Q
    TYPE(MATRIX) ::SIG_Q
    DOUBLE PRECISION ::NODAL_AREA
    INTEGER ::NUMBER_OF_NODES, NODE_NUMBER, I, UNIT_LENGTH,
   +                         DIV_PER_SIDE

    OPEN (UNIT = 1, FILE = 'FIELD_POINTS.TXT')
    OPEN (UNIT = 2, FILE = 'STRESS.TXT')
    OPEN (UNIT = 3, FILE = 'COORD_AND_FEM_STRESS_INPUT.TXT')

    READ (1,120)NUMBER_OF_NODES
    READ (1,*)NODAL_AREA
    READ (2,*)          !!SKIP THIS INFORMATION

    WRITE(3,*)NUMBER_OF_NODES

    PRINT*,"INPUT THE UNIT LENGTH PER SIDE OF THE FEM MODEL."
    READ*,UNIT_LENGTH
    WRITE(3,*)UNIT_LENGTH

    PRINT*,"INPUT THE NUMBER OF DIVISIONS PER SIDE OF THE FEM MODEL."
    READ*,DIV_PER_SIDE
    WRITE(3,*)DIV_PER_SIDE

    DO I=1, NUMBER_OF_NODES

      READ(1,*)NODE_NUMBER, Q%V(1), Q%V(2), Q%V(3)
      READ(2,100)SIG_Q%V(1)%V(1), SIG_Q%V(2)%V(2), SIG_Q%V(3)%V(3),
   +                         SIG_Q%V(1)%V(2), SIG_Q%V(2)%V(3), SIG_Q%V(1)%V(3)

            SIG_Q%V(2)%V(1) = SIG_Q%V(1)%V(2)
            SIG_Q%V(3)%V(2) = SIG_Q%V(2)%V(3)
            SIG_Q%V(3)%V(1) = SIG_Q%V(1)%V(3)


            WRITE(3,110)NODE_NUMBER,Q%V(1), Q%V(2), Q%V(3), SIG_Q%V(1)%V(1),
   +              SIG_Q%V(1)%V(2), SIG_Q%V(1)%V(3), SIG_Q%V(2)%V(1),
   +                         SIG_Q%V(2)%V(2), SIG_Q%V(2)%V(3), SIG_Q%V(3)%V(1),
   +                         SIG_Q%V(3)%V(2), SIG_Q%V(3)%V(3)
           END DO

!10    FORMAT(I5,3(3X,E12.5))
100   FORMAT(9X,6(ES12.5))
110   FORMAT(I5,1X,3(ES12.5,1X),9(ES12.5,1X))
120   FORMAT(I15)

           PRINT*,"TOTAL NUMBER OF NODES PROCESSED:",NUMBER_OF_NODES
           PRINT*,"HAVE A NICE DAY."
    END PROGRAM COORDINATES_AND_STRESS
!==================================================================!
```

APPENDIX D: ANSYS Batch Code

***Note:** This batch code enables the user to implement an ANSYS analysis without the time and trouble of manually building a model, meshing it, applying materials properties etc. Through the use of this code, the analysis becomes automatic. The reversed tractions calculated by Microplasticity are sent to NERSC by way of FTP. Once the "read information from" command is initiated and the file march_test.log is keyed in, the tractions are automatically loaded onto the FEM model and the analysis is run.

march_test.log

```
! /COM,ANSYS RELEASE  5.4    UP19971215         11:54:27
08/26/1999
/input,start,ans
,/usr/local/pkg/usg/ansys54/docu/,,,,,,,,,,,,,,,1
!*
/INPUT,model,log,/u/jkt/martinez/,1,0
/INPUT,alconst,log,/u/jkt/martinez/,1,0
/INPUT,FEM_OUTPUT,TXT,/u/jkt/martinez/,1,0
/INPUT,sol,log,/u/jkt/martinez/,1
```

model.log

```
/COM,ANSYS RELEASE  5.4    UP19971215         15:27:40
06/23/1999
/input,start,ans
,/usr/local/pkg/usg/ansys54/docu/,,,,,,,,,,,,,,,1
/UNITS,SI
/PREP7
DOF,ROTX,ROTY,ROTZ
BLOCK,0,2.85e-6,0,2.85e-6,0,2.85e-6,
!* march 9
ET,1,SOLID45
!*
UIMP,1,EX, , ,78.5e9,
UIMP,1,DENS, , ,, 7.86,
UIMP,1,ALPX, , , ,
UIMP,1,REFT, , ,300,
UIMP,1,NUXY, , , ,
UIMP,1,PRXY, , ,0.25
UIMP,1,GXY, , ,,
UIMP,1,MU, , , ,
UIMP,1,DAMP, , , ,
UIMP,1,KXX, , , ,
UIMP,1,C, , , ,
UIMP,1,ENTH, , , ,
UIMP,1,HF, , , ,
UIMP,1,EMIS, , , ,
UIMP,1,QRATE, , , ,
UIMP,1,MURX, , , ,
UIMP,1,MGXX, , , ,
UIMP,1,RSVX, , , ,
UIMP,1,PERX, , , ,
UIMP,1,VISC, , , ,
UIMP,1,SONC, , , ,
!*
ESIZE,0,10,
MSHAPE,0,3D
MSHKEY,1
!*
CM,_Y,VOLU
VSEL, , , ,       1
CM,_Y1,VOLU
CHKMSH,'VOLU'
CMSEL,S,_Y
!*
VMESH,_Y1
!*
CMDEL,_Y
CMDEL,_Y1
CMDEL,_Y2
!*
FINISH
!*
```

alconst.log

```
/PREP7
!*
FLST,2,1,1,ORDE,1
FITEM,2,967
D,P51X,,,,,,ALL
!*
FINISH
```

FEM_OUTPUT.TXT

```
 FLST,2,1,1,ORDE,1
 FITEM,2,  1
 F,P51X,FX, -1.449388058299793E-008
 FLST,2,1,1,ORDE,1
 FITEM,2,  1
 F,P51X,FY,  9.875644418140620E-009
 FLST,2,1,1,ORDE,1
 FITEM,2,  1
 F,P51X,FZ, -1.550973272305620E-008
 FLST,2,1,1,ORDE,1
 FITEM,2,  2
 F,P51X,FX,  1.550973272305620E-008
 FLST,2,1,1,ORDE,1
 FITEM,2,  2
 F,P51X,FY,  9.875644418140620E-009
 FLST,2,1,1,ORDE,1
 FITEM,2,  2
 F,P51X,FZ,  1.449388058299793E-008
 FLST,2,1,1,ORDE,1
```

Etc….

sol.log

```
/POST1
/SOLU
! /STAT,SOLU
SOLVE
FINISH
/GRAPHICS,FULL
/POST1
AVPRIN,0,0,
!*
! PRNSOL,S,COMP
/OUTPUT,ANSYS_STRESS,TXT,/u/jkt/martinez/,APPEND
PRNSOL,COMP !keep?
/OUTPUT     !keep?
FINISH
```

APPENDIX E: MATERIALS PROPERTIES

Iron

Lattice constant = 2.87×10^{-10} m [38, p.4-160]
Mu = 36.4×10^{9} Pa [40]
NU = 0.25 [40]
Peierls ratio 1 to 2 and 1 to 10
Peierls threshold 1 x10^{-3} Mu [1]
Density = 7.86 Mg/m^3 [40]

Copper

Lattice constant = 3.615×10^{-10} m [38, p.4-160]
Mu = 27×10^{9} Pa[Fivel] 43.1×10^{9} Pa [40]
NU = 0.35 [40]
Peierls ratio 1 to 2 ?
Peierls threshold 1×10^{-6} Mu [1]
Density = 8.94 Mg/m^3 [40]

Tantalum

Lattice constant = 3.30x 10^{-10} m [38, p.12-18]
Mu = 68.9×10^{9} Pa [40]
NU = 0.35 [40]
Peierls ratio ?
Peierls threshold 0.0×10^{-6} Mu [ ]
Density = 16.8 Mg/m^3 [40]

REFERENCES

1. D. Hull, D. J. Bacon, Introduction to Dislocations 3rd Edition, Butterworth Heinemann, Oxford, 1998
2. J. P. Hirth, J. Lothe, Theory of Dislocations, MacGraw-Hill Book Company, New York, 1968
3. J. A. Collins, Failure of Materials in Mechanical Design 2nd Edition, Wiley-Interscience Publications, New York, 1993
4. O. Khalfallah, M. Condat, L. Priester, "Image Force on a Lattice Dislocation Due to a Grain Boundary in B.C.C. Metals", *Philosophical Magazine A*, Vol. 67, No. 1, 231-250,1993
5. Theory of Crystal Defects, Gruber edition, Proceedins of the Summer School in Hrazary, Academic Press, New York, 1966
6. F. R. Nabarro, Report of a Conference on Strength of Solids, University of Bristol, The Physical Society, London, 1948
7. R. D. Cook, Finite Element Modeling for Stress Analysis, John Wiley & Sons Inc., New York, 1995
8. C. Teodosiu, Elastic Models of Crystal Defects, Springer-Verlag, 1982
9. S. N. Rosenbloom , C. Laird, "Fatigue Crack Nucleation Based on a Random Slip Process-I Computer Model", *Acta Metall. Mater.*,Vol. 41, No.12, 3473-3482, 1993

10. E. A. Repetto, M. Ortiz, "A Micromechanical Model of Cyclic Deformation and Fatigue-Crack Nucleation in F.C.C. Single Crystals", *Acta Mater.*,Vol. 45, No. 6, 2577-2595, 1997

11. Z. S. Basinski, R. Pascual, S. J. Basinski, "Low Amplitude Fatigue of Copper Single Crystals-I. The Role of the Surface in Fatigue Failure", *Acta Metall.*,Vol. 31, No. 4, 591-602, 1983

12. B. T. Ma, C. Laird, "Overview of Fatigue Behavior in Copper Single Crystals-I. Surface Morphology and Stage I Crack Initiation Sites for Tests at Constant Strain Amplitude", *Act Metal.*, Vol. 37, No. 2, 325-336, 1989

13. X. H. Liu, F. M. Ross, K. W. Schwarz, "Dislocated Quantum Dots", IBM Watson Research Center, Yorktown Heights, New York, 10598, 1999

14. S. D. Gavazza, D. M. Barnett*, J. Mech. Phys. Solids*, 24, 171, 1976 [need??]

15. N. M. Ghoniem, "Computational Methods For Mesoscopic, Inhomogeneous Plastic Deformation", Proceedings of the 1st. Latin American Summer School on Materials Instabilities, Valparaiso, Chile, Kluwer Publishing, Nov. 30-Dec. 4,1998

16. T.H. Alden, W.A. Backofen, "The Formation of Fatigue Cracks in Aluminum Single Crystals, *Acta Metallurica*, Vol. 9, 352-366, April, 1961

17. C. Coupeau, J. Grilhe, "Quantitative Analysis of Surface Effects of Plastic Deformation", *Materials Science and Engineering* A271, 242-250, 1999

18. M. Verdier, M. Fivel, I. Groma, “Mesoscopic scale simulation of Dislocation Dynamics in FCC metals: Principles and Applications*”, Modelling Simul. Sci. Eng.*, Vol. 6, 755-770, 1998

19. C.F. Robertson, M. Fivel, “A Study of the Submicron Indent-Induced Plastic Deformation”, *J. Mater. Res.*,Vol. 14, No. 6, 2251-2258, June, 1999

20. N. Urabe, J. Weertman, “Dislocation Mobility in Potassium and Iron Single Crystals”, *Materials Science and Engineering*, Vol. 18, 41-49, 1975

21. S. D. Wang, Sanboh Lee, “Screw Dislocations Near a Subsurface Crack in a Thin Plate”, *Journal of Materials Science and Engineering* A, Vol. 246, 61-68, 1998

22. Juan a. Hurtado, Kyung-Sur kim, “Scale Effects in Friction of Single Asperity Contacts. I. From Concurrent Slip to Single-Dislocation-Assisted slip*”, Proc. R. Soc. Lond.* A, Vol. 455, 3363-3384, 1999

23. Quanfang Chen, Greg P. Carmen, “Microscale Tribology (Friction) Measurement and the Influence of Crystal Orientation and Fabrication Process”, To be Published, 2000

24. G. E. Beltz, L.B. Freund, “Dislocation Threading Through an Epitaxial Film: An analysis Based on the Peierls-Nabarro Concept*”, Materials Research Society Symposium Proceedings* Vol. 308, Materials Research Society, Pittsburg, Pennsylvania, 1993

25. Jesper Vejlo Carstensen, Ph.D. Thesis: “Structural Evolution and Mechanisms of Fatigue in Polycrystalline Brass”, Riso National Laboratory, Roskilde, Denmark, March, 1998

26. Juan a. Hurtado, Kyung-Sur kim, “Scale Effects in Friction of Single Asperity Contacts. II. Multiple-Dislocation-Cooperated Slip”, *Proc. R. Soc. Lond*. A, Vol. 455, 3385-3400, 1999

27. E. B.Tadmor, R. Miller and R. Phillips, “Nanoindentation and Incipient Plasticity”, *J. Mater. Res*., Vol. 14, No.6, 2233-2249, June, 1999

28. John Hutchinson, “Plasticity at the Micron Scale*”, International Journal of the Solids and Structures*, Vol. 37, Issue 1-2, 225-238, 2000

29. Nasr Ghoniem and R. J. Amodeo, *Sol. State Phenom*. Vol. 3&4, 377, 1988

30. J Lepinoux and L. P. Kubin, *Scr. Met*., Vol. 21, No.33, 1987

31. G. H. Campbell et al, “Multiscale Modeling of Polycrystal Plasticity: A Workshop Report”, *Materials Science and Engineering* A251, 1-22, 1998

32. LLNLWeb page: http://www.gov/str/Moriarty.html “Predicting Material Behavior from the Atomic Level Up”, 2000

33. H. Mughrabi, “A Two-Parameter description of Heterogeneous Dislocation Distributions in Deformed Metal Crystals”, *Materials Science and Engineering*, Vol.85, 15-81, 1987

34. M. C. Fivel and A. A. El-Azab, “Linking Continuum Mechanics and 3-D Discrete Dislocation Simulations”, *J. Phys. IV***,** Vol. 9, 261-270, 1999

35. M. C. Fivel, T. J. Gosling, G. R. Canova, “Implementing Image Stresses in a 3D Dislocation Simulation”, *Modeling Simul. Mater. Sci. Eng*., Vol. 4, 581-596, 1996

36. C. Lemarchand, J. L. Chaboche, B. Devincre, L.P. Kubin, “Multiscale Modeling of Plastic Deformation”, *Journal De Physique IV*, Vol. 9, 271-277, 1999

37. M. Tang, l. P. Kubin, G. R. Canova, "Dislocation Mobility and the Mechanical Response of BCC Single Crystals: A Mesoscopic Approach", *Acta Mater.*, Vol. 46, No. 9, 3221-3235, 1998

38. D. R. Lide, CRC Handbook of Chemistry and Physics, 71st Edition, CRC Press, Boston, 1990-1991

39. J. Friedel, Dislocations, Pergamon press, New York, 1964

40. J. J. Tuma, Handbook of Physical Calculations, 2nd Edition, McGraw-Hill Book Company, New York, 1983

41. F. Nabarro, Theory of Crystal Dislocations, Clarendon Press, London, 1967

42. J. Menn, "IBM Delivering World's Fastest Computer to Livermore Lab", Los Angeles Times, A3, June 29, 2000

43. B. Bhushan, J. Israelachvili, U. Landman, "Nanotribology: Friction, Wear and Lubrication at the Atomic Scale", *Nature*, Vol. 374, April, 1995

44. E. van der Giessen, A. Needleman, "Discrete Dislocation Plasticity: A Simple Planar Model", *Modelling and Simulation in Materials Science and Engineering*, Vol. 3, No. 5, 689-735, 1995

45. J. Boussinesq, Application des Potentiels a L'Equilibre et du Mouvement des Solides Elastiques, Gauthier-Villars, Paris, p.45,1985

46. E. Tadmor, The Quasicontinuum Method, Ph.D. Thesis, Brown University, 1996

47. M. Daw, M. Baskes, *Phys. Rev. Lett.*, Vol. 50, 1285, 1983

48. J.J. Gilman, *Philosophical Magazine*, Series 8, Vol. 6, 159, 1961

49. J.J. Gilman, W. G. Johnston, *Journal of Applied Physics*, Vol. 31, 632, 1960

50. S. Amelinckx, The Direct Observation of Dislocations, Academic Press, New York, 1964

51. N. M. Ghoniem, S. H. Tong, l. Z. Sun, "Parametric Dislocation Dynamics: A Thermodynamics-Based Approach to Investigations of Mesoscopic Plastic Deformation", *Physical Review B*, Vol. 61, Num. 2, 913-927, 2000

52. T. Hughes, Computational Methods for Transient Analysis, North-Holland, Amsterdam, 1983

53. E. Suhir, Materials Research Society Symposia Proceedings: Heteroepitaxy On Silicon II, "Stresses In Multilayered Thin Films On A Thick Substrate", Materials Research Society, Pittsburg, PA, 73-80, 1987