A NUMERICAL INVESTIGATION OF THE EFFECT OF INDUCED POROSITY ON THE ELECTROMECHANICAL SWITCHING OF FERROELECTRIC CERAMICS

By

MORGAN BROWN

A thesis submitted in partial fulfillment of

the requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

WASHINGTON STATE UNIVERSITY

School of Mechanical and Materials Engineering

December 2005

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation/thesis of MORGAN BROWN find it satisfactory and recommend that it be accepted.

_____

Chair

_____

_____

ACKNOWLEDGMENTS

I first wish to thank my wife and family for their unending support in this venture.

I wish to thank my advisor at Washington State University Dr. Jow-Lian Ding for his sound advice, attention to details, and the deep knowledge of this subject matter which he has graciously shared with me.

Thanks are also due to Josh Robbins, my advisor at Sandia, for his superb computer science teaching skills, patience, and ability to clarify the most technical and complex of problems with well grounded logic.

I also wish to thank Dr A. Bandyopadhyay and Dr. S. Bose for their valuable input in regards to this thesis and its revisions.

Thanks are also due to Sandia National Laboratories for both funding and technical support.

NUMERICAL INVESTIGATION OF THE EFFECT OF INDUCED POROSITY ON THE ELECTROMECHANICAL SWITCHING OF FERROELECTRIC CERAMICS

Abstract

By Morgan Brown, M.S.
Washington State University
December 2005

Chair: Jow-Lian Ding

Ferroelectric ceramics have many applications ranging from microelectromechanical system (MEMS) to explosively driven power supplies. Besides chemical compositions and processing methods, porosity is an important material parameter that can affect both the electrical and mechanical responses of a ferroelectric. The main objective of the current study is to gain some preliminary insight on the possible effect of porosity on the switching behavior of ferroelectrics.

Numerical simulation is used to address the research objective. The numerical code used is an arbitrary Lagrangian-Eulerian, multi-material, multi-physics finite element code developed by Sandia national Laboratories. A phenomenological electromechanical model developed by Landis is implemented in the code first. The effects of void density which ranges from 0 to 11% and shape which includes sphere versus cylinder are then investigated through parametric study.

The study indicates that the remanent polarization decreases with the increased porosity density. For a given density, the porous solid that contains the cylindrical voids whose longitudinal axis is perpendicular to the applied electric field contains the largest amount of the remanent polarization. The solid that contains the cylindrical voids whose longitudinal axis is parallel to the applied electric field has the least remanent polarization.

The solid containing spherical voids possesses intermediate polarization. The possible reason for the geometry effect is that the void diverge the electric field and the polarization along the applied electric field direction. Different shapes of void result in different diverging effect.

The limitation of the current numerical simulation and possible future work are also discussed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Ferroelectric refers to a class of piezoelectric material which possesses spontaneous electric dipoles and the dipoles can be reoriented along certain specific crystallographic axes by an electric and/or mechanical field. A well-known example of ferroelectric material is lead zirconate titanate (PZT). Due to the random distribution of electric dipoles both within a grain and a polycrystal, the net or remanent polarization of a raw ferroelectric is essentially zero. Thus an important step in processing ferroelectrics is poling. In this process, an electric field is applied to align or switch the dipoles towards the applied electric field as much as possible. After a ferroelectric material is poled, it can then be applied in various applications ranging from microelectromechanical system (MEMS) to explosively driven power supplies. More detailed background on piezoelectricity can be found in Ref.1. Besides material and device development, numerous efforts have also been made to develop a constitutive model for the electromechanical behavior of ferroelectrics. The model is essential for predicting the material responses under complex electromechanical loadings. For model development, two general approaches are typically used. One is a phenomenological approach and the other is a microelectromechanical approach. For the former, perhaps the earliest effort is the work done by Chen et al. [2,3]. More recent efforts can be found in the work by Landis [4]. For the latter, an example can be found in the work by Huber et al [5]. A review of the constitutive models for ferroelectrics can also be found in [6]. For calibration and verification of the model, electromechanical switching experiments are typically used [7-8].

Besides chemical compositions and processing methods, porosity is an important material parameter that can affect both the electrical and mechanical responses of a ferroelectric. Porosity could result from the inherent defects in the material and electromechanical overloading. It could also be intentionally induced in the material as a way to tailor the material properties. For example, in the work by Tuttle et al. [9], the effect of porosity on phase transformation pressure for PZT was investigated using PZT with controlled porosity. The porosity was induced in the material using organic pore former. There have also been some theoretical studies on the porosity effect, but nearly all of them have been centered on the effective linear electromechanical properties of PZT containing voids. Examples of this study can be found in the work by Dunn and Taya [10,11] and Li et al. [12] among others.

The main objective of the current study is to gain some preliminary insight on the possible effect of porosity on the switching behavior of ferroelectrics. As mentioned earlier, poling is an important step of processing ferroelectric. For a porous ferroelectric, this step determines the initial distribution of electric dipoles and the remanent polarization, i.e. the initial condition of the material. Thus it is a critical issue from the practical point of view. The approach to be used in this study is numerical simulation. The effects of both density and shape of the voids on switching are investigated.

2. APPROACH

As mentioned earlier, numerical simulation is used to address the current research objective. The numerical code used is ALEGRA which is an arbitrary Lagrangian-Eulerian, multi-material, multi-physics finite element code developed by Sandia national Laboratories. An introduction of this code can be found in [13]. For the current study, ALEGRA is used to solve the electrostatic equations, namely,

$$D_{i,i} = q^v, \qquad (2.1)$$

and

$$e_{ijk}E_{j,k} = 0 \text{ or } E_i = -\phi_{,i} \qquad (2.2)$$

The boundary condition is given by

$$\|D_i\|\, n_i = q^s \qquad (2.3)$$

where D_i is the electric displacement, q^v is the free charge per unit volume, E_i is the electric field, ϕ is the electric potential, e_{ijk} is the permutation symbol, and n_i is the unit normal to the surface.

In order to study the effect of porosity on switching, an appropriate switching model has to be implemented in the code. The details of the model and its implementation will be described in the next section. After the model is implemented, a parametric study is then performed to investigate the effects of density and void shape on the switching behavior of a ferroelectric.

The preprocessing software used for mesh generation is CUBIT also developed by Sandia National Laboratories. The post processing is done with the EnSight software developed by Computational Engineering International, Inc.

3. ELECTROMECHANICAL CONSTITUTIVE MODEL

As mentioned earlier, development of constitutive models has followed two general approaches, namely microanatomical and phenomenological. For the current research objective, the latter is most appropriate for two reasons. One is that a phenomenological continuum model allows us to concentrate on the effect of porosity without dealing with the detailed complexity of the switching mechanisms and the effects of crystalline grains such as grain size, orientation, and grain boundary. Secondly, a phenomenological model is computationaly much more efficient than a micromechanical model. Among the several phenomenological models available in the literature, the model developed by Landis [4,7] is the most recent one. In addition, the material parameters used in the model have also been calibrated with the experimental data reported in [14] . Hence, it was selected for the current study. To facilitate the following discussion, the model is briefly summarized in the following.

3.1 Landis's phenomenological electromechanical model

The formulation starts with a Helmholtz free energy in the following form:

$$\Psi = \Psi^s(\epsilon_{ij}, \epsilon^r_{ij}, D_i, P^r_i) + \Psi^r(\epsilon^r_{ij}, P^r_i) \qquad (3.1)$$

where Ψ^s represents the reversible or stored part of the free energy and Ψ^r represents the free energy associated only with the internal state of the material. D_i is the electric displacement, P^r_i the remnant polarization, $\boldsymbol{\epsilon}_{ij}$ the strain, and ϵ^r_{ij} the remnant strain. Derivatives of Ψ with respect to ϵ_{ij} and D_i lead to the stress and electric field respectively, i.e.

$\sigma_{ij} = \frac{\partial \Psi^s}{\partial \epsilon_{ij}}$, and $E_i = \frac{\partial \Psi^s}{\partial D_i}$.Furthermore, derivative of Ψ with respect to ϵ^r_{ij} and P^r_i leads to the their energy conjugates as follows.

$$\widehat{\sigma}_{ij} = \frac{\partial \Psi}{\partial \epsilon^r_{ij}} = -\sigma_{ij} - \overline{\sigma}_{ij} + \sigma^B_{ij}, \qquad (3.2)$$

and

$$\widehat{E}_i = \frac{\partial \Psi}{\partial P^r_i} = -E_i - \overline{E}_i + E^B_i \qquad (3.3)$$

where $\sigma^B_{ij} = \frac{\partial \Psi^r}{\partial \epsilon^r_{ij}}$ is the back stress, $E^B_i = \frac{\partial \Psi^r}{\partial P^r_i}$ is the back electric field, $-\overline{\sigma}_{ij}$ is the difference between $\widehat{\sigma}_{ij}$ and $-\sigma_{ij}+\sigma^B_{ij}$ and $-\overline{E}_i$ is the difference between $\widehat{E}_i$ and $-E_i+E^B_i$. With ϵ^r_{ij}, P^r_i, $\widehat{\sigma}_{ij}$, and $\widehat{E}_i$ identified, a dissipative potential can be defined as

$$\Phi\left(\hat{\sigma}_{ij}, \hat{E}_i, \epsilon^r_{ij}, P^r_i\right) = 0 \qquad (3.4)$$

From Φ and the associated rule and consistency relation used in plasticity, the following relations can be derived,

$$\dot{\epsilon}^r_{ij} = \lambda \frac{\partial \Phi}{\partial \hat{\sigma}_{ij}} \qquad (3.5)$$

$$\text{and } \dot{P}^r_i = \lambda \frac{\partial \Phi}{\partial \hat{E}_{ij}} \qquad (3.6)$$

where $\lambda = \frac{1}{D}\left(\hat{\epsilon}_{ij}\dot{\sigma}_{ij} + \hat{P}_i\dot{E}_i\right)$ is the plastic multiplier and the expressions for D, $\hat{\epsilon}_{ij}$, and $\hat{P}_i$ are detailed in [4]. Furthermore,

$$\dot{\sigma}^B_{ij} = H^m_{ijkl}\dot{\epsilon}^r_{kl} + H^{em}_{kij}\dot{P}^r_k \qquad (3.7)$$

$$\text{and } \dot{E}^B_i = H^{em}_{ikl}\dot{\epsilon}^r_{kl} + H^e_{ij}\dot{P}^r_j \qquad (3.8)$$

where

$$H^m_{ijkl} = \frac{\partial^2 \Psi^r}{\partial \epsilon^r_{ij}\partial \epsilon^r_{kl}} \qquad (3.9)$$

$$H^{em}_{kij} = \frac{\partial^2 \Psi^r}{\partial P^r_k \partial \epsilon^r_{ij}} \qquad (3.10)$$

$$H^e_{ij} = \frac{\partial^2 \Psi^r}{\partial P^r_i \partial P^r_j} \qquad (3.11)$$

The above equations set up the general framework of the model. The specific functional forms used in the model are the following.

$$\Psi^s(\epsilon_{ij}, \epsilon^r_{ij}, D_i, P^r_i) =$$
$$\frac{1}{2}c^D_{ijkl}\left(\epsilon_{ij} - \epsilon^r_{ij}\right)\left(\epsilon_{kl} - \epsilon^r_{kl}\right) - h_{kij}\left(D_k - P^r_k\right)\left(\epsilon_{ij} - \epsilon^r_{ij}\right) + \frac{1}{2}\beta^\epsilon_{ij}\left(D_i - P^r_i\right)\left(D_j - P^r_j\right),$$

(3.12)

where c_{ijkl}, h_{kij}, and β_{ij} are the electromechanical moduli;

$$\Psi^r(\epsilon^r_{ij}, P^r_i) = \Psi^m(\epsilon^r_{ij}) + \Psi^e(P^r_i) =$$
$$\tfrac{1}{2}H^m_o\epsilon_c\left[\tfrac{J^e_2}{\epsilon_c}\exp\left(\tfrac{m_m}{1-\frac{\bar{\epsilon}}{\epsilon_c}}\right)\right]^2 + H^e_o P^2_o\left[\ln\left(\tfrac{1}{1-\frac{P^r}{P_o}}\right) - \tfrac{P^r}{P_o}\right] \qquad (3.13)$$

$$\text{where } \bar{\epsilon} = J^e_2 f\left(\tfrac{J^e_3}{J^e_2}\right),\ J^e_2 = \left(\tfrac{2}{3}e^r_{ij}e^r_{ij}\right)^{\frac{1}{2}},\ J^e_3 = \left(\tfrac{4}{3}e^r_{ij}e^r_{jk}e^r_{ki}\right)^{\frac{1}{3}} \qquad (3.14)$$

$$\text{with } e^r_{ij} = \epsilon^r_{ij} - \tfrac{1}{3}\epsilon^r_{kk}\delta_{ij}, \qquad (3.15)$$

$$f\left(\tfrac{J^e_3}{J^e_2}\right) = -0.0965\left(\tfrac{J^e_3}{J^e_2}\right)^3 + 0.01\left(\tfrac{J^e_3}{J^e_2}\right)^6 + 0.8935 \qquad (3.16) \qquad \text{for } \tfrac{J^e_3}{J^e_2} > 0$$

and

$$f\left(\tfrac{J^e_3}{J^e_2}\right) = -0.1075\left(\tfrac{J^e_3}{J^e_2}\right)^3 - .027\left(\tfrac{J^e_3}{J^e_2}\right)^6 - 0.028\left(\tfrac{J^e_3}{J^e_2}\right)^{21} + 0.8935\tfrac{J^e_3}{J^e_2} \qquad (3.17) \text{ for } \tfrac{J^e_3}{J^e_2} \geq 0;$$

$$\Phi = \tfrac{\hat{E}_i\hat{E}_i}{E^2_o} + \tfrac{3\hat{s}_{ij}\hat{s}_{ij}}{2\sigma^2_o} + \tfrac{\beta(\hat{s}_{ij}E^r_iP^r_j + \hat{s}_{ij}E^r_jP^r_i)}{2E_oP_o\sigma_o} - 1 = 0 \qquad (3.18)$$

$$\text{where } \hat{s}_{ij} = \hat{\sigma}_{ij} - \tfrac{1}{3}\sigma_{kk}\delta_{ij}. \qquad (3.19)$$

With the above functional forms incorporated, a compact form of the constitutive model can be cast as follows.

$$\dot{\epsilon}_{ij} = \left(s^E_{ijkl} + \tfrac{1}{D}\widehat{\epsilon}_{ij}\widehat{\epsilon}_{ij}\right)\dot{\sigma}_{kl} + \left(d_{kij} + \tfrac{1}{D}\widehat{P}_k\widehat{\epsilon}_{ij}\right)\dot{E}_k \qquad (3.20)$$

$$\dot{D}_i = \left(d_{ikl} + \tfrac{1}{D}\widehat{P}_i\widehat{\epsilon}_{kl}\right)\dot{\sigma}_{kl} + \left(\kappa^\sigma_{ij} + \tfrac{1}{D}\widehat{P}_i\widehat{P}_j\right)\dot{E}_j \qquad (3.21)$$

where s^E_{ijkl}, d_{ikl}, κ^σ_{ij} are the elastic compliance, piezoelectric, and dielectric permittivity tensors respectively, and given in the following forms.

$$s^E_{ijkl} = \tfrac{1+\nu}{2Y}\left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right) - \tfrac{\nu}{Y}\delta_{ij}\delta_{kl} \qquad (3.22)$$

$$d_{kij} = \tfrac{P^r}{P_o}[d_{33}n_kn_in_j + d_{31}n_k\alpha_{ij} + \tfrac{1}{2}d_{15}\left(n_i\alpha_{jk} + n_j\alpha_{ik}\right)] \qquad (3.23)$$

$$\kappa^\sigma_{ij} = \kappa\delta_{ij} \qquad (3.24)$$

where Y is Youngs modulus, ν is Poisson's ratio, $P^r = \sqrt{P^r_iP^r_i}$, $n_i = \frac{P^r_i}{P^r}$ and $\alpha_{ij} = \delta_{ij} - n_in_j$. Notice that the elastic compliance and dielectric permittivity tensors are treated as isotropic and the piezoelectric tensor is treated as transversely isotropic.

The material properties used in the current study follow those used in [7]. For convenience and continuity of the paper, the relevant material properties are listed in Table 1.

Table 3.1: Material parameters input for PZT

Symbol	Units	Value	Description
E_o	$\frac{V}{m}$	$0.35e6$	Coercive Field
P_o	$\frac{C}{m^2}$	0.3133	Polarization Corresponding to Coercive Field
σ_o	$\frac{N}{m^2}$	$2.75e7$	Stress corresponding to Coercive Field
d_{33}	$\frac{c}{N}$	$6.0e-10$	Piezoelectric Constant in the 33 direction
d_{31}	$\frac{c}{N}$	$-3.0e-10$	Piezoelectric Constant in the 31 direction
d_{15}	$\frac{c}{N}$	$9.0e-10$	Piezoelectric Constant in the 15 direction
κ	$\frac{c^2}{Nm^2}$	$3.0e-8$	Dielectric Permittivity
Y	Pa	$7.0e10$	Youngs Modulus
ν		0.4	Poisons Ratio

3.2 Implementation of the Model into ALEGRA

ALEGRA has the capability to incorporate external material models. In the current study, the material model was written in FORTRAN77. A detailed instruction on implementation is described in Appendix A. There are three major components needed to interface the material module with ALEGRA, namely, an interface file written in C++ and named as mclan.C, a header file named mclan.h, and a FORTRAN file for the material model named nonlin.F. A brief introduction of the structure and the function of these files are given in the following. All these files are fully annotated and included in Appendix A.

3.2.1. **Nonlin.F (APPENDIX A1)**

The original Landis's model is in index notation form and the model involves the operation between several high order tensors. A key step to programing this model in a easily tractable manner is to change the index form into a matrix form. As an example, one equation used in the model is in the following index notation form.

$$A_{kij}P_k = \frac{\partial d_{qij}}{\partial P_k^r}E_qP_k = [\frac{d_{33}}{P_o}(n_in_j\delta_{qk}+n_jn_q\delta_{ik}-2n_in_jn_qn_k)+\frac{d_{31}}{P_o}(\delta_{ij}\delta_{qk}+n_in_j\delta_{qk}-$$

$$n_in_q\delta_{jk}-n_jn_q\delta_{ik}+2n_in_jn_qn_k)+$$

$$\frac{d_{15}}{2P_o}(\delta_{ik}\delta_{jq}+\delta_{iq}\delta_{jk}-2n_jn_q\delta_{ik}-2n_in_q\delta_{jk}-2n_in_q\delta_{ik}-2n_in_j\delta_{qk}+4n_in_jn_qn_k)]E_qP_k \qquad (2.24)$$

This form can be changed into a matrix from as follows.

$$\frac{d_{33}}{P_o}[(\underline{E}*\underline{P})\underline{nn}+(\underline{n}*\underline{E})\underline{Pn}+(\underline{n}*\underline{E})\underline{nP}-2(\underline{E}*\underline{n})\underline{P}*\underline{n})\underline{nn}]+\frac{d_{31}}{P_o}[(\underline{E}*\underline{P})\underline{I}-$$

$$(\underline{E}*\underline{P})\underline{nn}-(\underline{n}*\underline{E})\underline{nP}-\underline{n}*\underline{E})\underline{Pn}+2(\underline{E}*\underline{n})(\underline{P}*\underline{n})\underline{nn}]$$

$$+\frac{d_{15}}{2P_o}[\underline{PE}+\underline{EP}-2(\underline{n}*\underline{E})\underline{Pn}-2(\underline{n}*\underline{E})\underline{nP}-2(\underline{E}*\underline{P})\underline{nn}+4(\underline{E}*\underline{n})(\underline{P}*$$

$$\underline{n})\underline{nn}] \qquad (2.25)$$

Since ALEGRA can only accept 77 version of FORTRAN, a series of functions and subroutines need to be created and declared as external in order to perform some of the mathematical requirements of the model. They include:

Table 3.2: Functions

FUNCTION **NAME**	**RETURNS**
NORM	the magnitude of a vector
TRACE	trace of a vector
DOTV	the dot product of two vectors
SIGNORM	the dot product of a vector to a second order tensor to a vector

Table 3.3: Subroutine Names

SUBROUTINE NAME	**this returns the output "O" in the form:**
$CTEN1(C,V,O)$	$O_i = cV_i$
$CTEN1(C,T,O)$	$O_{ij} = cT_{ij}$
$VDOTT(V,T,O)$	$O_i = V_jT_{ij}$
$TDOTT(T1,T2,O)$	$O_{ij} = T_{ik}T_{kj}$
$TSUM(T1,T2,O)$	$O_{ij} = T_{ij}+T_{ij}$
$DYADV(V1,V2,O)$	$O_{ij} = V_iV_j$
$INIV(V,O)$	$O_i = 0*V_i$
$INITT(T,O)$	$O_{ij} = 0*T_{ij}$
$VEQV(V1,V2,O)$	$O_i = V2_i$
$TEQT(T1,T2,O)$	$O_{ij} = T2_{ij}$

3.2.2. Mclan.h (APPENDIX A2)

This header file is used to set up the variable functions and dictate array sizes which will be used in the malcn.C file. As this file is basically a set of subclasses of the ALEGRA Material_Model class, it must conform to strict guidelines. These guidelines allow the ALEGRA code to understand the material model and include such requirements as the ErrorStatus functions, which are set simply as a boolean. The file starts with a set of include statements which must be set for the pre-processor to link this file appropriately. Like all C++ classes there is both a public and private section. First the public section is entered, where the input data variables are declared. Each of these will correspond to the value which can be assigned by the user in order to specify the type of material desired. Next the ErrorStatus functions are declared, including Compute_Lab_Constants, Set_Up, Initialize_State and Update_State. These are required by the Material_Model class of which this is a subclass. Each of these will be called in the mclan.C file and will be described in detail there. This completes the public section.

The private section deals with variables needed only internally by this material model. Constants must be declared here which are used to set the size of vectors used for storage purposes in the mclan.C file. Variable id's are also set as well as the input, output and state variables such as electric field, stress strain and electric displacement.

3.2.3 Mclan.C (APPENDIX A3)

This file creates the initial setup and storage of the material model, as well as associated variables necessary to such. First the includes are set for preprocessing, which link this back to the mclan.h file among others. At the start the external FORTRAN file must be

declared, along with its desired size. Following this the user specified input variables are declared, and their values assigned. These must appear in the same order and include the same variables as both the mclan.h and input files.

The first function called is the Set_Up function. Here the element variables such as electric field, stretch and rotation are dealt with. Next the material variables must be registered, these include those which are specific to this model such as electric displacement.

The second function called is Initialize_State. Here the storage for data, vectors and tensors are initialized in size to be used in subsequent sections. The values which will not change from element to element are also set at this point. This is done to increase the computation speed by removing the necessity to compute them for each subsequent iteration. The vectors and tensors, such as the remnant polarization, which will be passed into the FORTRAN subroutine are initialized to zero here. Once the FORTRAN subroutine is called each incremental change will be applied to these vectors and tensors. The Compute_Lab_Constants function is also called here. This is done in order to create the permittivity tensor for the initial state. The bulk and shear moduli are also set at this point. Alegra requires the permittivity, bulk and shear modulus to be set at this point, and will crash if this step is not carried out.

Finally the Update_State function is called. This is the main body of the model. Mesh generated values computed by ALEGRA, such as the current electric field are assigned and their saved values from the previous steps are then used to determine incremental values. All the variables needed by the Landis model must be specified explicitly here. These values are then passed to the FORTRAN subroutine, nonlin.F, which creates the incremental

values of electric displacement and strain and pass them back. These are then summed with their current values and saved. The final step in this function is to save the current value of the electric field so that it can be used in the following step to determine the increment of electric field.

3.3 Build the executable

Once the 3 files described in section 3.2 are complete, Alegra must be compiled with these files as a designated material model. It is important to place them in a directory appropriate to their use and to set the proper links to this area. In this case the alegra/materialmodels/electromechanical directory. Then ALEGRA must be re-compiled with the options desired, in this case for debugging purposes the dbg option was added. The full extent of this step is detailed in Appendix C. Once this has been accomplished an alegra/Build directory will have been created. Entering this directory and typing "make" will re-make any new files which have been added, allowing changes to be made to the existing files.

4. RVE BOUNDARY CONDITIONS, ELECTRIC INPUT, AND VALIDATION OF THE MODEL

The focus of the current study is the electric switching as occurred in poling. Thus only electric field is applied and Eqns. 3.20 and 3.21 can be simplified to

$$\dot{\epsilon}_{ij} = \left(d_{kij} + \tfrac{1}{D}\widehat{P}_k\widehat{\epsilon}_{ij}\right)\dot{E}_k \qquad (4.1)$$

$$\dot{D}_i = \left(\kappa^{\sigma}_{ij} + \tfrac{1}{D}\widehat{P}_i\widehat{P}_j\right)\dot{E}_j \qquad (4.2)$$

The numerical simulation is performed on a 2 cm x 2 cm x 2 cm representative volume element (RVE). The volume was meshed with 3D hexagoal elements using CUBIT, mentioned earlier. The simulation geometry is shown in Figure 4.1. A 0.02 Hz, $\pm$1.08e6V/m sinusoidal electric field is applied to the two sides normal to the z direction and periodic boundary conditions are applied to the remaining four surfaces. The periodic boundary conditions translate the properties of the nodes on the surface ABFE to the corresponding ones on DCGH and from BFGC to AEHD.

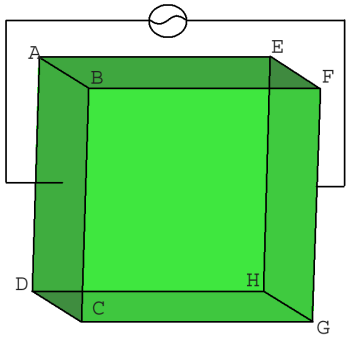
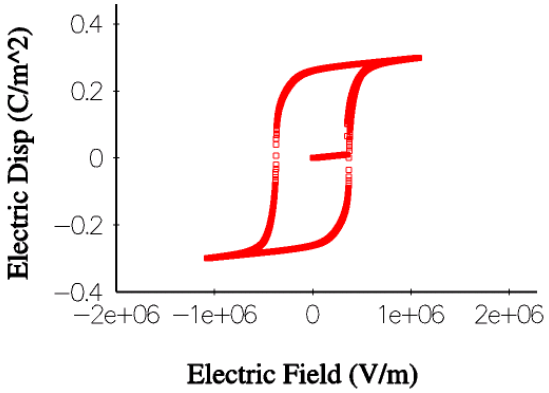


Figure 4.1: Simulation configuration and periodic boundary conditions.

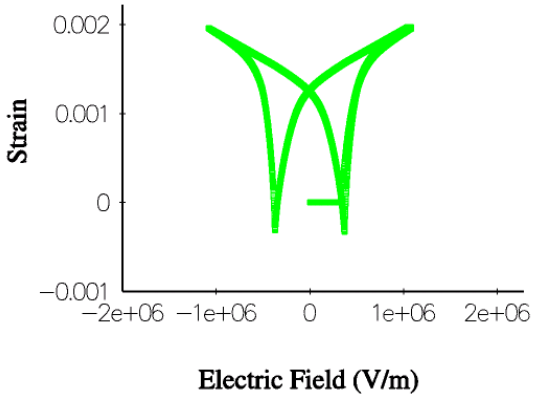
The output from CUBIT is a genesis file, the mclan.gen file. This is part of the input for ALGERA. Besides mclan.gen, an input file mclan.inp is also needed. The input

file contains the information on material properties, boundary conditions, forcing input and output control etc. Once the physical geometry has been created, the input material characteristics set and the material model implemented in ALEGRA can be called. This is accomplished by setting a path to the ALEGRA executable, the details of which are located in Appendix C, and then simply using the command "Alegra run-id" . This will create a series of output files. The key one is the exodus file, mclan.exo in this case, which contains the output data. The post processing software, ENSIGH8, is then used to analyze the output. Shown in Figure 4.2 are the hysteresis and butterfly loops generated by ALEGRA. These loops compare well with those presented in [4,7] and serve as a validation of the model implemented in the code.

The physical basis of the hysteresis loop is the following. The sample is initially unpoled. As the electric field is applied, only linear piezoelectric effect is induced initially without any increase in remnant polarization. When the electric field reaches the coercive field, switching begins and remnant polarization begins to accrue, sharply at first with a very small increase in electric field resulting in a great increase in polarization. The back electric field, remains small at this point. Then as the remnant polarization begins to reach the state of lock-up the back electric field begins to grow rapidly and rate of polarization decreases correspondingly as a result of the kinematic hardening. When the electric field decreases, the material responds linearly to the electric unloading until the switching criterion is met again. The continuous decrease of electric field then leads to the decrease of remanent polarization.



a)



b)

Figure 4.2: (a) Hysteresis and (b) butterfly loops.

5. EFFECT OF POROSITY

In this section, the effect of void density and shape on the switching behavior of PZT is investigated. The outputs of the simulations include remanent polarization, electric displacement, strain and electric field. Two types of voids are investigated, namely, sphere and cylinder. The simulation configurations are shown in Figure 5.1. For the cylindrical void, two types of electric loading are studied as shown in Figures (b) and (c). In Figure (b), the electric field is perpendicular to the longitudinal axis of the cylinder, while in Figure (c) it is parallel. The meshes used in this study are shown in Figure 5.2 (a) and (b) for the spherical and cylindrical voids respectively. The total number of elements used are ~35k for (a) and (b). This number is a result of convergence study as will be shown later. As the area near the pore is of specific interest to this study, all meshes were graded with mesh going from fine near the pore to course at the surfaces. This was accomplished by first webcutting the block, into the pore and other regions favorable to meshing. The entire volume was then merged to guarantee mesh congruency between regions. The surface of the pore was then meshed using the "pave command" which Automatically meshes a surface with an unstructured quadrilateral mesh. This mesh was then swept from the pore to the external faces, with a positive bias, in the form of Hexagonal elements. The pore was then meshed with hexahedral elements as well. Smoothing techniques were implemented and the final mesh then examined using mesh quality checks built into the Cubit software package. The files used to created each of these can be found in Appendix B2.

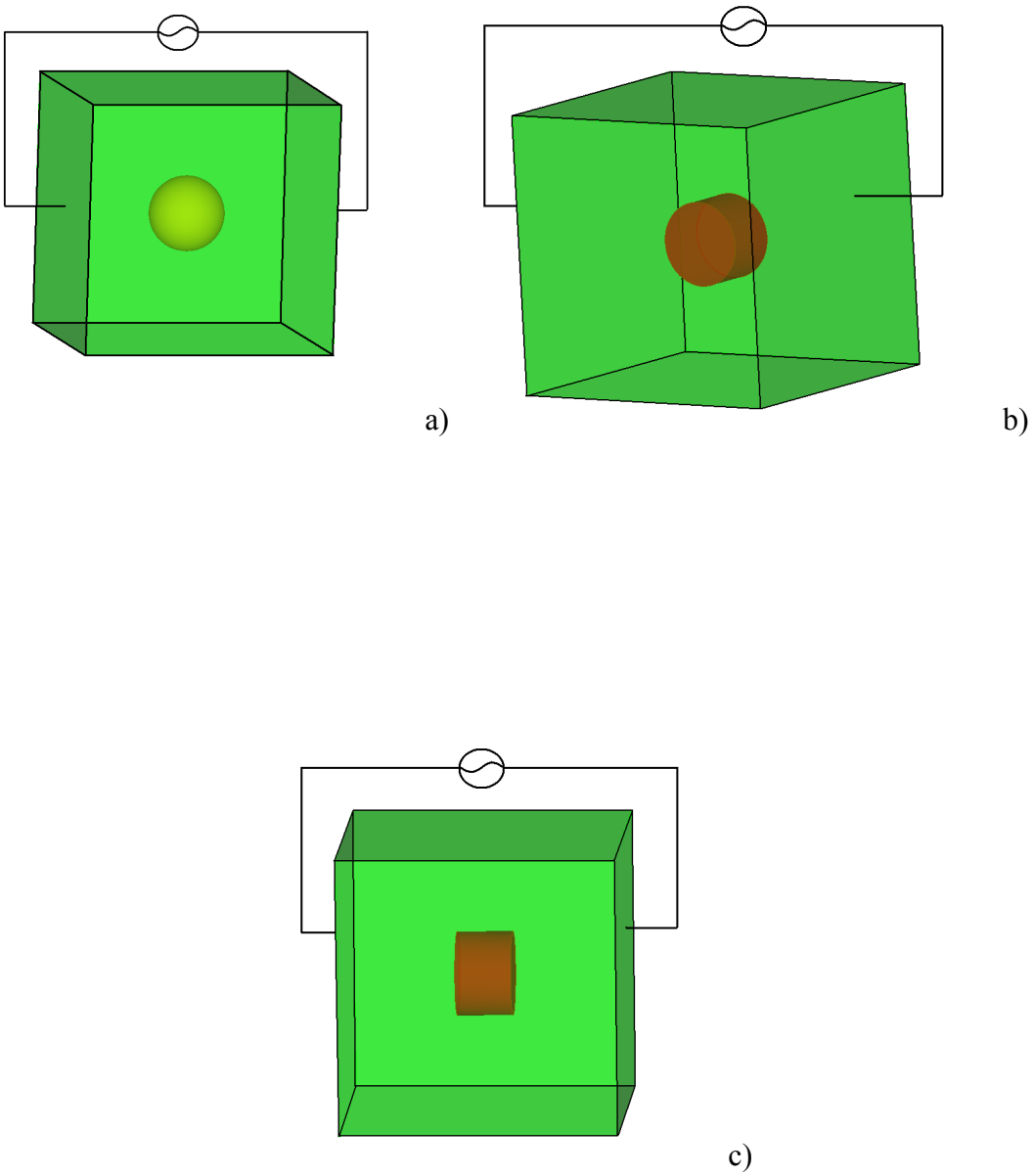


Figure 5.1: RVE for spherical void (a), cylindrical void with longitudinal axis perpendicular to the applied electric field (b), and cylindrical void with longitudinal axis parallel to the applied electric field (c).

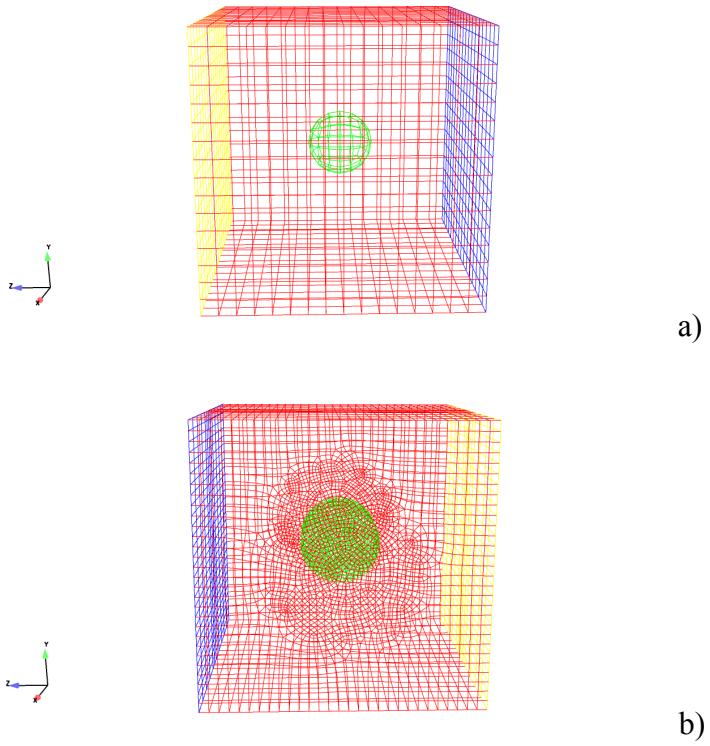


Figure 5.2: Meshed RVE for (a) spherical void, and (b) cylindrical void.

5.1 Effect of Void Density for Spherical Void

In this section, the density effect for a spherical void on the switching behavior is investigated. Void density is the ratio of the void volume with the volume of the solid. Three different void densities are studied, namely, 1.4%, 6.5%, and 11%. All the other conditions including applied potential magnitude, boundary conditions and mesh density were kept constant for all the simulations. The overall electric displacement and remanent polarization for the porous aggregate along the applied electric field direction can be calculated by integrating P_i^r along the applied electric field direction over the volume and then divide the integral by the volume of the RVE.

The hysteresis loops corresponding to different spherical void densities are shown in Figure 5.3. The loop corresponding to the flawless solid as shown in Figure 4.2 is also included here as a reference. A plot of the remanent polarization as a function of void density is shown in Figure 5. 4 indicating that the remanent polarization decreases with the increased void density.

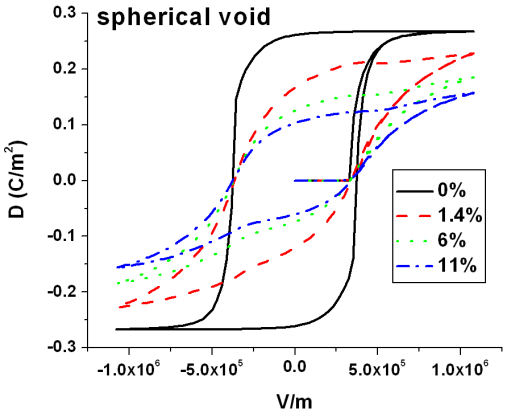


Figure 5.3: Aggregate polarization vs. electric field hysteresis plots for 3 spherical void volume fractions and solid.

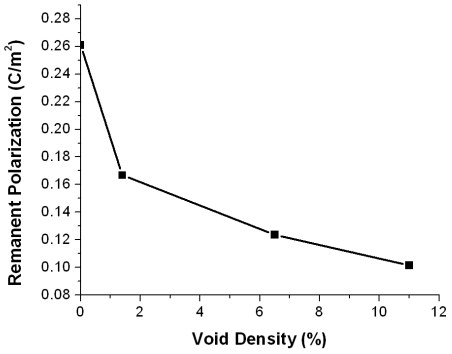


Figure 5.4: Trend line depicting maximum polarization values with varying porosity.

5.2 Effect Void Shape

The hystereses loops for porous ferroelectric with cylindrical voids are shown in Figures 5.5 (b) and (c). Similar to the spherical void, the remanent polarization decreases with the increased void densities. A collective plot of all the voids and densities is shown in Figure 5.6. An important observation from this figure is that the porous ferroelectric with cylindrical void whose longitudinal axis is perpendicular to the applied electric field possesses the largest remanent polarization, the solid with cylindrical void whose longitudinal axis is parallel to the applied electric field possesses the least, and the solid with spherical void is in the middle.

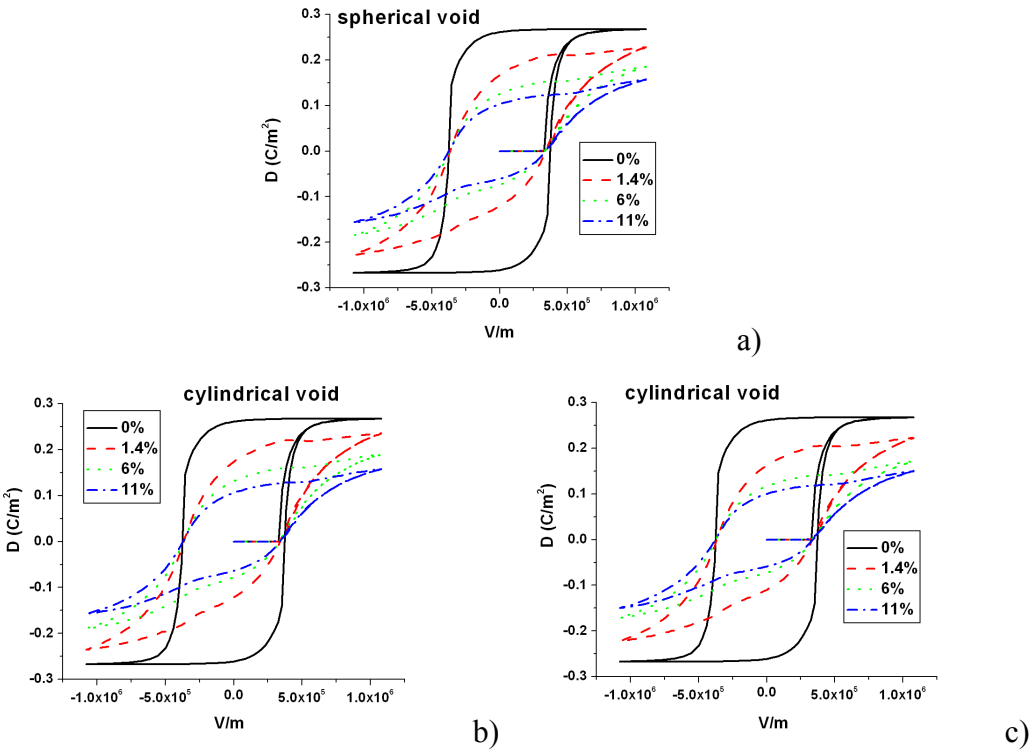


Figure 5.5: Effect of void density on hysteresis loop, (a) spherical void, (b)cylindrical void with its longitudinal axis perpendicular to the electric field, (c) cylindrical void with its longitudinal axis parallel to the electric field

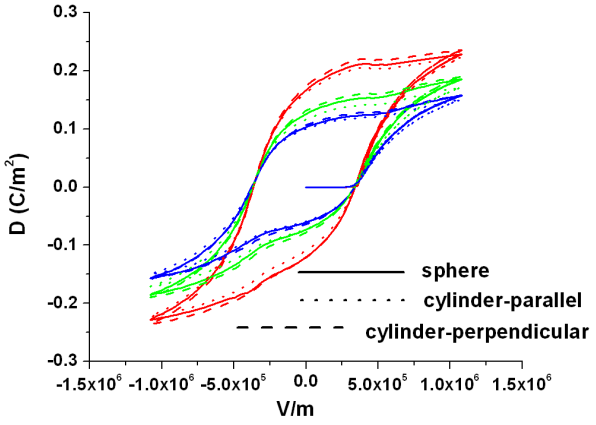


Figure 5.6: Effect of void density and shape on hysteresis loops

To ensure numerical convergence, the remanent polarization as a function of the number of elements used is studied and the results are shown in Figures 5.7 for the solids with 11% spherical and cylindrical voids respectively. All meshes were graduated with the best resolution at the pore. The figure shows that the results converges reasonably well after the number of elements is greater than 27K.

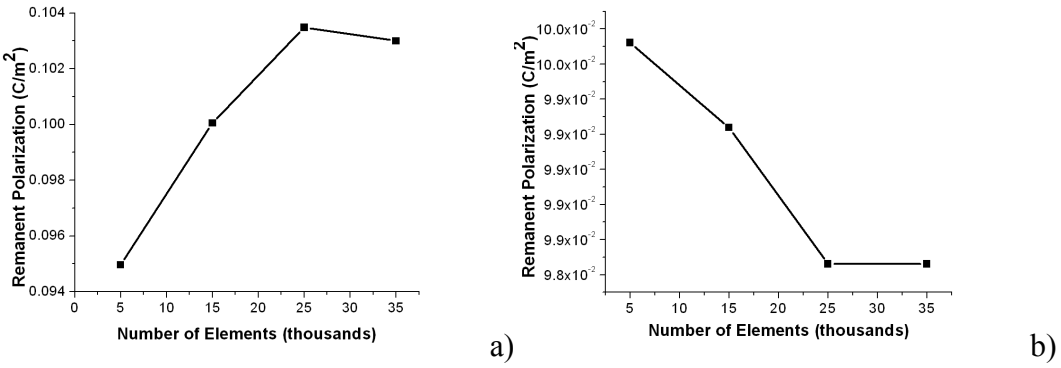
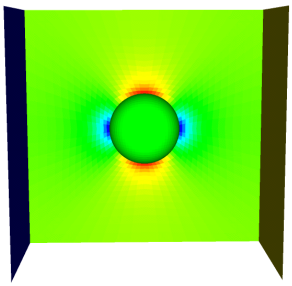


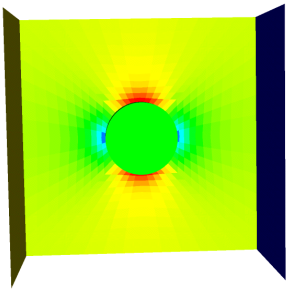
Figure 5.7: Remanent polarization versus the number of elements used in the simulation, (a) spherical void, (b)cylindrical void with its longitudinal axis perpendicular to the electric field.

6. DISCUSSION

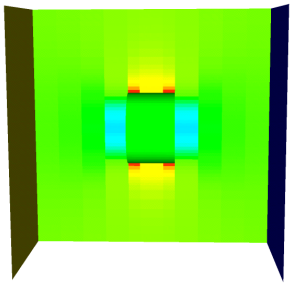
In order to gain some insights into the results shown in section 5, further investigation is carried out to study the distribution of electric field and polarization. Shown in Figure 6.1 is the distribution of the electric field for the solids with different types of porosity when the electric field is at its maximum. The magnitude of the electric field that is perpendicular to the surface of the pore (left and right side of the pore) is significantly lower than that tangential to the surface (top and bottom side of the pore). This distribution is a result of the electrostatic boundary conditions. The corresponding distribution of the magnitude of the remanent polarization on the center plane of the material is shown in Figure 6.2. The distributions shown in Figures 6.1 and 6.2 are similar to each other.



a)



b)



c)

Figure 6.1: Distribution of the electric field for a solid with (a) spherical void, (b)cylindrical void with its longitudinal axis perpendicular to the electric field, (c) cylindrical void with its longitudinal axis parallel to the electric field

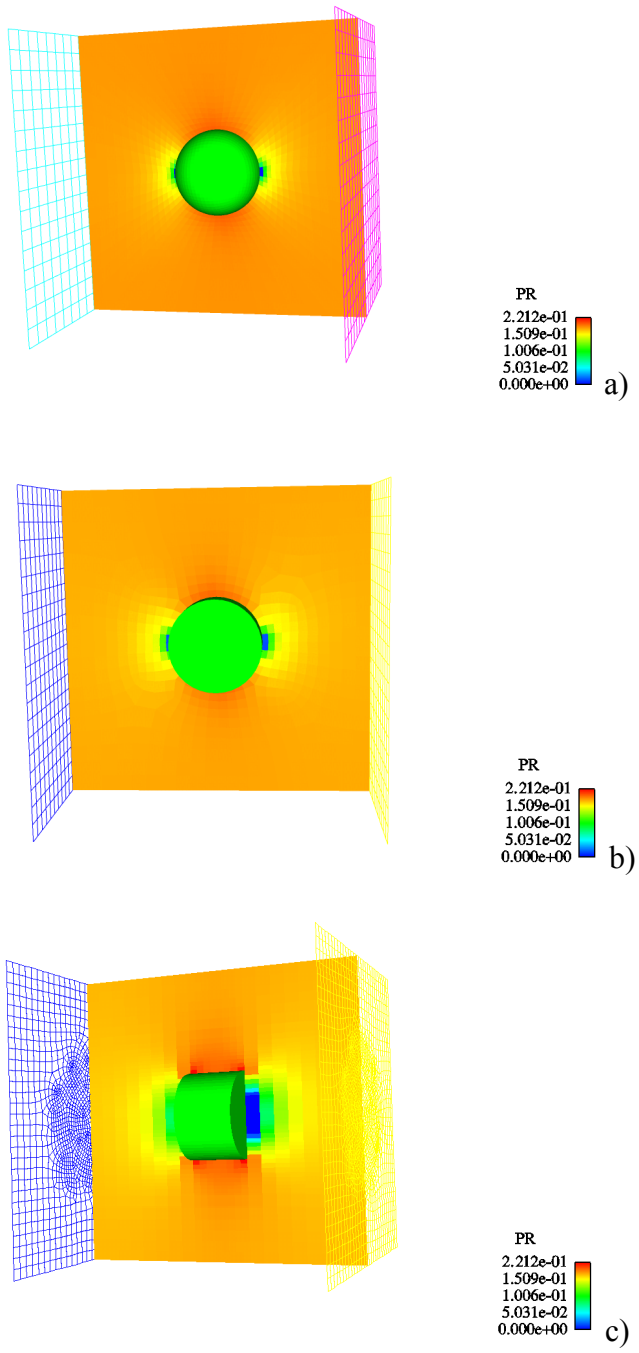
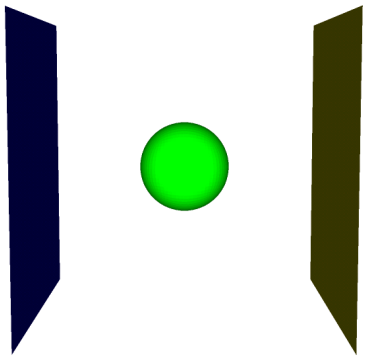
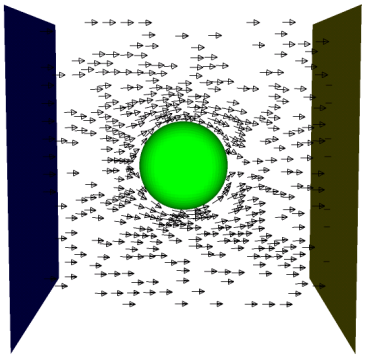


Figure 6.2: Distribution of the remanent polarization for a solid with (a) spherical void, (b)cylindrical void with its longitudinal axis perpendicular to the electric field, (c) cylindrical void with its longitudinal axis parallel to the electric field

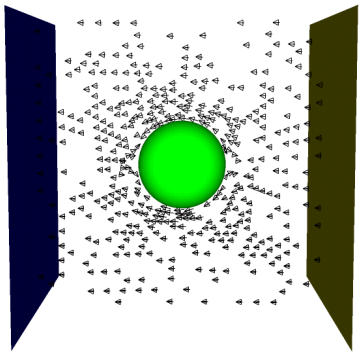
More insights can also be gained by the distribution of the electric displacement and remanent polarization vectors. Shown in Figure 6.3 is the distribution of the electric displacement for a solid containing spherical voids during the switching process. Each element has a polarization vector, however in order to get a clear view of vector distribution, only 30% of the vector population is plotted. The length of the vector is proportional to the magnitude of the polarization. The solid is unpolarized initially. As the electric field increases, the electric displacement starts to develop. They circles around the pore and their amplitude increases with the applied electric field. The electric displacement on the sides of the pore parallel to the applied potential difference has smaller amplitude. When the electric field reverses, the electric displacements in this area are the last ones to switch. The electric displacements on the top and bottom of the pore have the largest amplitude. These are also the areas where the switching starts first.



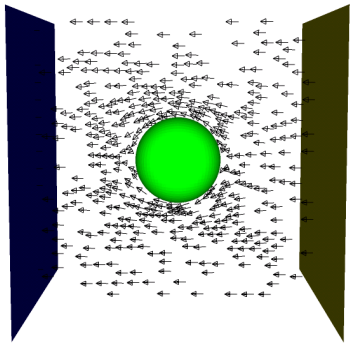
a)



b)



c)

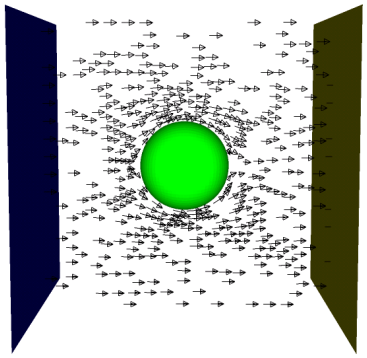


d)

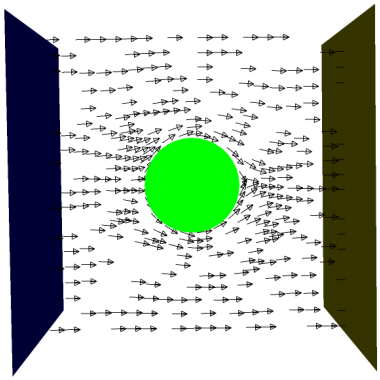
Figure 6.3: Distribution of the electric displacement during switching for a solid with spherical void. (density = 30%)

The distribution of the remanent polarization vectors are shown in Figure 6.4. The vectors can be seen to diverge around the pore. For the solid with cylindrical void whose

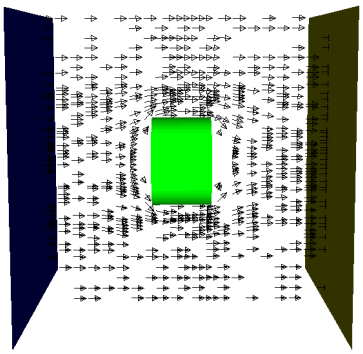
longitudinal axis is perpendicular to the electric field, both the electric field and polarization only have in-plane diverging. However, for the other two types of voids, out-of-plane divergence exists. The solid with cylindrical void whose longitudinal axis is parallel to the electric field appears to have the largest the out of plane divergence which in turn leads to the lowest remanent polarization. This appears to be the main reason for the observed void shape effect.



a)



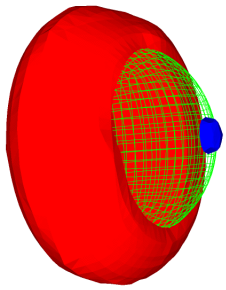
b)



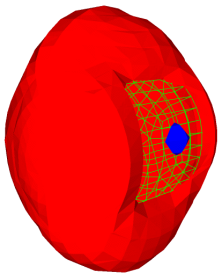
(c)

Figure 6.4: Distribution of the remanent polarization vector for a solid with (a) spherical void, (b)cylindrical void with its longitudinal axis perpendicular to the electric field, (c) cylindrical void with its longitudinal axis parallel to the electric field

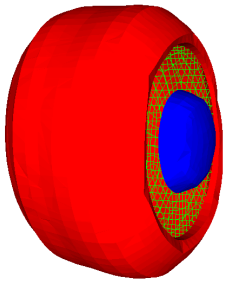
The distribution of the remanent polarization can also be seen with the iso-surface plot as shown in Figure 6.5 where red color represents a polarization of .2 and blue represents a polarization of .02 C/m^2. The figure indicates that red surface for the solid whose cylindrical void is perpendicular to the applied electric field direction covers the largest volume around the void. For the solid whose cylindrical void is parallel to the applied electric field direction, the volume covered by the red surface is the least because little area is covered along the direction of the electric field. The solid with the spherical void is somewhere in between.



a)



b)



c)

Figure 6.5: Iso-surface plots of the remnant polarization in the direction of applied electric field for a solid with (a) spherical void, (b)cylindrical void with its longitudinal axis perpendicular to the electric field, (c) cylindrical void with its longitudinal axis parallel to the electric field. The red color indicates a surface of value .2 (C/m^2) and the blue surface indicates a surface of .02 (C/m^2) .

7. POSSIBLE FUTURE WORK

To the author's knowledge, the current work is probably the first theoretical attempt to address the effects of porosity on the switching behavior of ferroelectric ceramic. The results obtained appear to be reasonable. As mentioned earlier, poling establishes the initial conditions of ferroelectric ceramic. With the initial conditions properly established, many other issues could be studied such as the coupling effects between electric and mechanical loadings, and effects of various electromechanical loading paths etc. As far as the numerical simulation is concerned, one critical issue is the solution of the stress field. ALEGRA is an explicit shock dynamics code. It solves the stresses corresponding to a given strain field. However, during poling, the electric displacement, total strain, remnant polarization, remnant strain and the corresponding residual stresses have to be solved iteratively. This may require a significant modification of the current code.

It has also been noted that slightly advanced switching is present along the diagonal of the material. This issue has been extensively tested with a great variety of mesh options, including a variety of cut plane options but with no success. It has also been noted that when creating meshes, the smoothness of the meshes plays a crucial role in obtaining reasonable solutions. This issue appears to be related to the hardening terms in the model. More study is needed to gain more insights into the model.

8. CONCLUSION

In the current study, numerical simulation has been used to investigate the effects of void density and shape on the switching behavior of ferroelectric ceramic. Two types of voids were examined, namely sphere and cylinder. The void density ranges from 0 to 11%. It was found that the remanent polarization decreases with the increased porosity density. For a given density, the porous solid that contains the cylindrical voids whose longitudinal axis is perpendicular to the applied electric field contains the largest amount of the remanent polarization. The solid that contains the cylindrical voids whose longitudinal axis is parallel to the applied electric field has the least remanent polarization. The solid containing spherical voids possesses intermediate polarization. The possible reason for the geometry effect is that the void diverge the electric field and the polarization along the applied electric field direction. Different shapes of void result in different diverging effect.

REFERENCES

1. Jaffe, B., Cook, W., and Jaffe, H., "Piezoelectric Ceramics," Academic Press inc., 1971, New York.
2. Chen, P. J. and Montgomery, S. T., 1980, A macroscopic theory for the existence of the hysteresis and butterfly loops in ferroelectricity, *Acta Mechanica,* **23**, 199-208.
3. Chen, P. J., 1980, Three dimensional dynamic electromechanical constituive relations for ferroelectric materials, *Int. J. of Solids and Structures,* **16**, 1059-1067.
4. Landis, C.M., 2002. Fully coupled, multi-axial, symmetric constitutive laws for polycrystalline ferroelectric ceramics. *Journal of the Mechanics and Physics of Solids* **50**, 127-152.
5. Huber, J. E., Fleck, N. A., Landis, C. M., and McMeeking, R. M., 1999. A constituive model for ferroelectric polycrystals, *Journal of the Mechanics and Physics of Solids* **47**, 1663-1697.
6. Landis, C. M., 2004, Non-linear consitive modeling of ferroelectrics, *Cuurent Opinions in Solid State and Materials Sciences* **8**, 59-69.
7. Landis, C.M., Wang, J., Sheng, J., 2004. Micro-electromechanical determination of the possible remanent strain and polarization states in polycrystalline ferroelectrics and the implications for phenomenological constitutiv theories. *Journal of Intelligent Material Systems and Structures* **15**, 513-525.

8. Huber, J.E., Fleck, N.A., 2001. Multi-axial electrical switching of a feroelectric: theory versus experiment. *Journal of the Mechanics and Physics of Solids* **49**, 785-811.

9. Tuttle, B. A., Yang, P., Gieske, J. H. et al., 2001, Pressure-induced phase transformation of controlled porosity Pb(ZrTi)O ceramics, *Journal of the American Ceramics Sociey* **84(6)**, 1260-1264.

10. Dunn, M. L. and Taya, M., 1993, Electromechancial properties of porous piezoelectric ceramics, *Journal of the American Ceramics Cociety* **76**, 1697-1706.

11. Dunn, M. L. and Taya, M., 1993, Micromechancis predictions of the effective electroelastic moduli of piezoelectric composites, *Int. J. of Solids and Structures,* **30**, 161-175.

12. Li, Z., Wang, C. Chen, C., 2003. Effects of electromechanical properties of transversely isotropic piezoelectric ceramics with microvoids. *Computational Materials Science* 27, 381-392.

13. Summers, R. M., Peery, J. S., Wong, M. W., Hertel, E. S., Trucano, T. G., Chhabildas, L. C., 1997, Recent progress in ALEGRA devleopment and application to ballistic impacts, *Int. J. Impact Engng* 20, 779-788.

14. Lynch, C.S., 1996. The effect of uniaxial stress on the electro-mechanical response of 8/65/35 PLZT. *Acta Metallurgica* **10**, 4137-4148.

APPENDIX

Appendix A. IMPLEMENTED ALEGRA MODEL

A.1 Nonlin.F

This is the fortran subroutine called by the Mclan.C main file. The bulk of the material model has been implamented here.

Nonlin.F:

```
 SUBROUTINE NONLIN( EEXT,OEEXT,EEXTIN,SIG,SIGINC,
 * PR,EB,EPSR,SIGB,UI,
 * DINC,EPSINC)
C
 INCLUDE 'implicit.h'
 INCLUDE 'params.h'

C This is the switching model developed by Landis (2002)
 DIMENSION UI(*)
      DOUBLE PRECISION
 & EEXT(3),OEEXT(3),EDIR(3),EHT(3), EB(3), EBAR(3),
 & EEXTIN(3), EBINC(3), EHTOLD(3),
 & SIGHT(3,3), SIGB(3,3), SIGBAR(3,3), DEVHT(3,3),
 & SIGBIN(3,3), SIG(3,3), SIGINC(3,3), SHTOLD(3,3),
 & D(3), PR(3),DINC(3), PRIN(3), PTILD(3),
 & DINE1(3), DINE2(3),DINS1(3), DINS2(3),
 & PHT(3), PRDIR(3),PRMAG,
 & EPS(3,3), EPSR(3,3), EPSINC(3,3), EPSRIN(3,3),
 & EPINE1(3,3),EPINE2(3,3), EPINS1(3,3), EPINS2(3,3),
 & EPTILD(3,3), EPHT(3,3),EPRDEV(3,3),
 & AKIJPK(3,3), AIJPJ(3),UIJ(3,3),
 & UNIT(3,3),
 & PRPR(3,3), PTILPR(3,3),
 & PRPTIL(3,3), PTILEX(3,3), EXPTIL(3,3),
 & ESIGPR(3,3), SIGPRE(3,3), PRSGPR(3,3), SGPRPR(3,3),
 & SIGPR(3), EEXTPR(3,3), PREEXT(3,3),
 & ETAEP1(3,3), ETAEP2(3,3), ETAEP3(3,3),
 & TEMPZZ, DE, EOLD,ENEW,SIGOLD,SIGNEW,
 & TIME, DT, DDUM,
cc Additions for the nonlin elec part
 & DEDIR(3), EPSE1(3,3), EPSE2(3,3),
 & HIJ(3,3), ALFA(3,3), PRODNN(3,3),
 & HIJKP(3,3),HIJKEP(3),HIJKPEP,HMEP(3,3),
 & HMEPEP,DENOM,PLAST,
 & TEMP1,TEMP2,
 & TMP331(3,3),TMP332(3,3),TMP333(3,3),TMP334(3,3),
 & TMP335(3,3),TMP336(3,3),
 & TMP311(3),TMP312(3),
 & TMP1(3,3),TMP2(3,3),TMP3(3,3),TMP4(3,3),TMP5(3),
 & TMP6(3,3),TMP7(3,3),TEMPONE,
 & PHI,SIGNN,SIGNP,DOTNE,DOTPN,
cc
 & EXPME, EXPMT, EXPMC, EXPMPI,
 & CHII, H0SIG, H0PI,
```

```
cc Additions for nonlin mech part
 & TEMP3,TEMP5,EPSRNN,HIJKCOF,EPRI1,EPRJ2,EPRJ3,
 & RATIOJ,F,DF,DDF,
cc Dummy variables just used as temps
 & DUMC1,DUMC2,DUMC3,
 & DUMV1(3),DUMV2(3),DUMV3(3),DUMV4(3),DUMV5(3),DUMV6(3),
 & DUMT1(3,3),DUMT2(3,3),DUMT3(3,3),DUMT4(3,3),
cc for ALEGRA
 & E0,P0,SIG0,EP0,BTA,
 & D33,D31,D15,PERMI,
 & YOUNGS,POISS,SHMOD,AMDA,
 & PMAX,PMIN,PSAT,EPT,EPC,EXPMM,H0M,H0E
 EXTERNAL DEDOT, PERMEDOT, DSIGDOT, SSIGDOT,
 & NORM,TRACE,DOTV,
 & CTEN1,CTEN2,VDOTT,VSUM,TSUM,DYADV,
 & INITV,INITT,SIGNORM,
 & HIJKLEP, HMEPEP2
C Assign model specific values
      E0 = UI(1)
      P0 = UI(2)
      SIG0 = UI(3)
      D33 = UI(4)
      D31 = UI(5)
      D15 = UI(6)
      PERMI = UI(7)
      YOUNGS= UI(8)
      POISS = UI(9)
      EPC = 0.12e-2
      BTA = 2.95
      EXPMM = 0.01
      H0M = 1.375e7
      H0E = 0.175e5
c       EEXT: external electric field
c       EHT: E_height
c       EEXTIN: increment of electric vector field
c       PR: remanent polarization vector
c       PRIN: increment of PR
c       PRMAG: magnitude of remanent polarization vector
c       PRDIR: unit vector in the remanent polarization direction
c       SIG: applied stress tensor
c SIGHT: sigma_height
c       DEVHT: deviatoric stress_height
c SIGINC: increment of SIG
c       EPS: strain tensor
c       EPSR: remanent strain tensor
c EPSINC: increment of EPS
c EPSRIN: increment of EPSR
C calclulate EDIR, a unit vector in the direction of EEXT
      DUMC1= SQRT(EEXTIN(1)**2+EEXTIN(2)**2+EEXTIN(3)**2)
      DO 98 i=1,3
      EDIR(i)= EEXTIN(i)/DUMC1
```

```
98 CONTINUE
 TEMPZZ= 1/(SQRT(EDIR(1)**2+EDIR(2)**2+EDIR(3)**2))
 CALL CTEN1(TEMPZZ,EDIR,EDIR)
C       DERIVED VARIABLE
       SHMOD= YOUNGS/(2.*(1.+POISS))
       AMDA = (YOUNGS*POISS)/((1.+POISS)*(1.-2.*POISS))
c PSAT: defined in Landis (2004)


 PMAX=0.86*P0
 PMIN=PMAX
 EPT = EPC

c PSAT: defined in Landis (2004)
 PSAT= PMAX

C Create an identity matrix
       CALL INITT(UNIT,UNIT)
       UNIT(1,1)=1.
       UNIT(2,2)=1.
       UNIT(3,3)=1.
C SET INITIAL VALUES AS FOLLOWS
c PR: remanent polarization vector
c PRMAG: magnitude of remanent polarization vector
c PRDIR: unit vector in the remanent polarization direction
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C SIG: applied stress tensor
C SIGB: back stress
C SIGBAR: sigma_bar
C       SIG=0.
       CALL INITT(SIG,SIG)
C SIGINC: increment of SIG
C       SIGINC=0.
       CALL INITT(SIGINC,SIGINC)
C EB: back electric field
C EBAR: E_bar
C       EBAR=0
       CALL INITV(EBAR,EBAR)
C EPS: strain tensor
C EPSR: remanent strain tensor
C       EPS=0
       CALL INITT(EPS,EPS)
C SIGHT: sigma_height
C DEVHT: deviatoric stress_height
C       SIGHT=0.
       CALL INITT(SIGHT,SIGHT)
C       DEVHT=0.
       CALL INITT(DEVHT,DEVHT)
C EHT: E_height
C       EHT=0.
       CALL INITV(EHT,EHT)
```

```
C HERE THE INCREMENT OF ELECTRIC FIELD DE IS CREATED
      ENEW = SQRT(EEXT(1)**2+EEXT(2)**2+EEXT(3)**2)
      EOLD = SQRT(OEEXT(1)**2+OEEXT(2)**2+OEEXT(3)**2)
      DE = ENEW-EOLD
C       DEDIR=EDIR
 CALL VEQV(DEDIR,EDIR,DEDIR)
      TEMPONE = -1.0
      CALL CTEN1(TEMPONE,EDIR,DUMV1)
      IF(DE.LT.0.) CALL VEQV(DEDIR,DUMV1,DEDIR)
 PRMAG= SQRT(PR(1)**2+PR(2)**2+PR(3)**2)

 IF(PRMAG.EQ.0.) GO TO 12
      DO 99 i=1,3
      PRDIR(i)= PR(i)/PRMAG
99       CONTINUE
 GO TO 14
 12 CALL INITV(PRDIR,PRDIR)
 IF(DE.NE.0.) CALL VEQV(PRDIR,DEDIR,PRDIR)

C PRODNN: NN
C ALFA: EQN. (4.6)
 14 CALL DYADV(PRDIR, PRDIR, PRODNN)
C ALFA = UNIT-PRODNN
      CALL CTEN2(TEMPONE,PRODNN,DUMT1)
      CALL TSUM(UNIT,DUMT1,ALFA)
c DINC; increment of D
c PRIN: increment of PR
C       DINC =0.
 CALL INITV(DINC,DINC)
C       PRIN=0.
 CALL INITV(PRIN,PRIN)
c EPSINC: increment of EPS
c EPSRIN: increment of EPSR
C EPSINC=0.
      CALL INITT(EPSINC,EPSINC)
C EPSRIN=0.
      CALL INITT(EPSRIN,EPSRIN)
c EBINC: increment of EB
c SIGBIN: increment of SIGB
C EBINC=0.
      CALL INITV(EBINC,EBINC)
C SIGBIN =0.
      CALL INITT(SIGBIN,SIGBIN)
c EPINS2: increment of EP (epsilon: strain) due to the second term of sigma_dot (Eqn. 3.36)
c EPINE2: increment of EP (epsilon: strain) due to the second term of E_dot (Eqn. 3.36)
c DINS2: increment of D due to the second term of sigma_dot (Eqn. 3.37)
c DINE2: increment of D due to the second term of E_dot (Eqn. 3.37)
C EPINS2=0.
      CALL INITT(EPINS2,EPINS2)
C EPINE2=0.
      CALL INITT(EPINE2,EPINE2)
```

```
C DINS2=0.
      CALL INITV(DINS2,DINS2)
C DINE2=0.
      CALL INITV(DINE2,DINE2)

c EPINS1: increment of EP (epsilon: strain) due to the first term of sigma_dot (Eqn. 3.36)
c EPINE1: increment of EP (epsilon: strain) due to the first term of E_dot (Eqn. 3.36)
c DINS1: increment of D due to the first term of sigma_dot (Eqn. 3.37)
c DINE1: increment of D due to the first term of E_dot (Eqn. 3.37)
C EPINS1=0.
      CALL INITT(EPINS1,EPINS1)
C EPINE1=0.
      CALL INITT(EPINE1,EPINE1)
C DINS1=0.
      CALL INITV(DINS1,DINS1)
C DINE1=0.
      CALL INITV(DINE1,DINE1)

C CALL SSIGDOT(SIGINC, EPINS1)
 CALL SSIGDOT(SIGINC,POISS,YOUNGS, EPINS1)
C CALL DEDOT(EEXTIN, EPINE1)
 CALL DEDOT(EEXTIN,PRMAG,PMAX,PRDIR,D33,D31,D15,EPINE1)
C CALL DSIGDOT(SIGINC, DINS1)
 CALL DSIGDOT(SIGINC,PRDIR,PRMAG,PMAX,D33,D31,D15,DINS1)
C CALL PERMEDOT(EEXTIN, DINE1)
 CALL PERMEDOT(EEXTIN, PERMI,DINE1)
c EHTOLD: old (previous) EHT

C EHTOLD=EHT
      CALL VEQV(EHTOLD,EHT,EHTOLD)
 TEMP1=SIGNORM(SIG, PRDIR, PRDIR)
 TEMP2=DOTV(PRDIR, EEXT)
      CALL VDOTT(PRDIR,SIG,TMP311)
CCCCCCCC EBAR=
C here DUMV1-3 are dummy vectors
C here DUMC1-3 are dummy constants
      CALL CTEN1(TEMP1,EEXT,DUMV1)
      DUMC1 = -2.0*TEMP1*TEMP2
      CALL CTEN1(DUMC1,DUMV2,DUMV2)
      DUMC1 = 2.0*TEMP2
      CALL CTEN1(DUMC1,TMP311,DUMV3)
      CALL VSUM(DUMV1,DUMV2,DUMV1)
      CALL VSUM(DUMV1,DUMV3,DUMV1)
      DUMC1 = D33/PMAX
      CALL CTEN1(DUMC1,DUMV1,DUMV1)
      DUMC1 = TRACE(SIG)
      CALL CTEN1(DUMC1,EEXT,DUMV2)
      CALL CTEN1(TEMP1,EEXT,DUMV3)
      DUMC1 = TEMP1*-2.0*TEMP2
      CALL CTEN1(DUMC1,PRDIR,DUMV4)
      DUMC1 = -2*TEMP2
```

```
      CALL CTEN1(DUMC1,TMP311,DUMV5)
      CALL VSUM(DUMV2,DUMV3,DUMV2)
      CALL VSUM(DUMV2,DUMV4,DUMV2)
      CALL VSUM(DUMV2,DUMV5,DUMV2)
      DUMC1 = D31/PMAX
      CALL CTEN1(DUMC1,DUMV2,DUMV2)
      DUMC1 = 2.0
      CALL VDOTT(EEXT,SIG,DUMV3)
      CALL CTEN1(DUMC1,DUMV3,DUMV3)
      DUMC1 = -4.0*TEMP2
      CALL CTEN1(DUMC1,TMP311,DUMV4)
      DUMC1 = -2.0*TEMP1
      CALL CTEN1(DUMC1,EEXT,DUMV5)
      DUMC1 = -2.0*TEMP1*-2.0*TEMP2
      CALL CTEN1(DUMC1,PRDIR,DUMV6)
      CALL VSUM(DUMV3,DUMV4,DUMV3)
      CALL VSUM(DUMV3,DUMV5,DUMV3)
      CALL VSUM(DUMV3,DUMV6,DUMV3)
      DUMC1 = D15/(2.0*PMAX)
      CALL CTEN1(DUMC1,DUMV3,DUMV3)

      CALL VSUM(DUMV1,DUMV2,DUMV1)
      CALL VSUM(DUMV1,DUMV3,EBAR)
C EHT= EEXT-EB+EBAR
      DUMC1 = -1.0
      CALL CTEN1(DUMC1,EB,DUMV1)
      CALL VSUM(DUMV1,EBAR,DUMV1)
      CALL VSUM(DUMV1,EEXT,EHT)

c SHTOLD: old (previous) SIGHT
C SHTOLD=SIGHT
C       CALL TEQT(SHTOLD,SIGHT,SHTOLD)
CCCCCCC SIGHT=SIG-SIGB+SIGBAR
 CALL CTEN2(DUMC1,SIGB,DUMT1)
 CALL TSUM(DUMT1,SIGBAR,DUMT1)
 CALL TSUM(DUMT1,SIG,SIGHT)

CCCCCCC DEVHT=SIGHT-(TRACE(SIGHT)/3.)*UNIT
 DUMC1 = (-1)* TRACE(SIGHT)*(1/3.0)
 CALL CTEN2(DUMC1,UNIT,DUMT1)
 CALL TSUM(SIGHT,DUMT1,DEVHT)
CCCCCCCCCCC PHI=
 DUMC1= (EHT(1)**2+EHT(2)**2+EHT(3)**2)/E0**2
 CALL TDOTT(DEVHT,DEVHT,DUMT1)
 DUMC2= 3*TRACE(DUMT1)/(2.*SIG0**2)
 DUMC3= BTA*(SIGNORM(DEVHT, PR, EHT)+
 & SIGNORM(DEVHT,EHT,PR))
 & /(2*E0*SIG0*PMAX)-1.
 PHI = DUMC1+DUMC2+DUMC3

 IF (PHI.LE.0.) GO TO 1000
```

```
C
C Calculate epsilon_tilde (Eqn. 3.15) and P_tilde (Eqn. 3.16)
 CALL DYADV(EHT, PR, TMP331)
 CALL DYADV(PR, EHT, TMP332)
CCCCCCCCCCCC EPTILD=
      DUMC1 = 3.0*(1/(SIG0**2))
      CALL CTEN2(DUMC1,DEVHT,DUMT1)
      DUMC2 = BTA/(2.0*E0*PMAX*SIG0)
      CALL CTEN2(DUMC2,TMP331,DUMT2)
      CALL CTEN2(DUMC2,TMP332,DUMT3)
      DUMC1 = (-2.0/3.0)*DOTV(EHT,PR)*DUMC2
      CALL CTEN2(DUMC1,UNIT,DUMT4)
      CALL TSUM(DUMT1,DUMT2,DUMT1)
      CALL TSUM(DUMT1,DUMT3,DUMT1)
      CALL TSUM(DUMT1,DUMT4,EPTILD)
CCCCCCCCCCCCC PTILD =
      DUMC1 = 2.0/(E0**2)
      CALL CTEN1(DUMC1,EHT,DUMV1)
      DUMC2 = BTA/(E0*PMAX*SIG0)
      CALL VDOTT(PR,DEVHT,DUMV2)
      CALL CTEN1(DUMC2,DUMV2,DUMV2)
      CALL VSUM(DUMV1,DUMV2,PTILD)

C Calculate A_ij P_j (eqn. 3.33)
 SIGNN=SIGNORM(SIG,PRDIR, PRDIR)
 SIGNP=SIGNORM(SIG, PTILD, PRDIR)
 TEMP5 = DOTV(PRDIR,PTILD)

CCCCCCCCCCCCC
      DUMC1 = 2.*TEMP5
      CALL VDOTT(PRDIR,SIG,DUMV1)
      CALL VDOTT(DUMV1,ALFA,DUMV1)
      CALL CTEN1(DUMC1,DUMV1,DUMV1)
      DUMC1 = SIGNN
      CALL VDOTT(PTILD,ALFA,DUMV2)
      CALL CTEN1(TEMP5,PRDIR,DUMV3)
      CALL VSUM(DUMV2,DUMV3,DUMV2)
      CALL CTEN1(DUMC1,DUMV2,DUMV2)
      DUMC1 = D33/PMAX
      CALL VSUM(DUMV1,DUMV2,DUMV1)
      CALL CTEN1(DUMC1,DUMV1,DUMV1)
      DUMC1 = -2.*TEMP5
      CALL VDOTT(PRDIR,SIG,DUMV2)
      CALL VDOTT(DUMV2,ALFA,DUMV2)
      CALL CTEN1(DUMC1,DUMV2,DUMV2)
      CALL TDOTT(ALFA,SIG,DUMT1)
      DUMC1 = TRACE(DUMT1)
      CALL CTEN1(TEMP5,PRDIR,DUMV3)
      CALL VDOTT(PTILD,ALFA,DUMV4)
      CALL VSUM(DUMV3,DUMV4,DUMV3)
      CALL CTEN1(DUMC1,DUMV3,DUMV3)
```

```
      DUMC1 = D31/PMAX
      CALL VSUM(DUMV2,DUMV3,DUMV2)
      CALL CTEN1(DUMC1,DUMV2,DUMV2)
C D
      DUMC1 = -2*SIGNN
      CALL VDOTT(PTILD,ALFA,DUMV3)
      CALL CTEN1(DUMC1,DUMV3,DUMV3)
C C
      DUMC1 = -2*TEMP5
      CALL TDOTT(ALFA,SIG,DUMT1)
      CALL VDOTT(PRDIR,DUMT1,DUMV4)
      CALL CTEN1(DUMC1,DUMV4,DUMV4)
C B
      CALL TDOTT(SIG,ALFA,DUMT1)
      CALL VDOTT(PTILD,DUMT1,DUMV5)
      DUMC1 = 2.*DOTV(PRDIR,DUMV5)
      CALL CTEN1(DUMC1,DUMV5,DUMV5)
C A
      CALL TDOTT(SIG,ALFA,DUMT1)
      CALL TDOTT(ALFA,DUMT1,DUMT1)
      CALL VDOTT(PTILD,DUMT1,DUMV6)
      DUMC1 = 2.
      CALL CTEN1(DUMC1, DUMV6,DUMV6)
      CALL VSUM(DUMV3,DUMV4,DUMV3)
      CALL VSUM(DUMV3,DUMV5,DUMV3)
      CALL VSUM(DUMV3,DUMV6,DUMV3)
      DUMC1 = D15/(2.*PMAX)
      CALL CTEN1(DUMC1,DUMV3,DUMV3)
      CALL VSUM(DUMV1,DUMV2,DUMV1)
      CALL VSUM(DUMV1,DUMV2,DUMV1)
      CALL VSUM(DUMV1,DUMV3,AIJPJ)
C
C
C
C Calculate A_kij P_k (eqn. 3.32)

c PRPR : nn
c PTILPR: P_tilde n
c PRPTIL: n P_tilde
c PTILEX: P_tilde E
c EXPTIL: E P_tilde

 CALL DYADV(PTILD, PRDIR, PTILPR)
 CALL DYADV(PRDIR, PTILD, PRPTIL)
 CALL DYADV(PTILD, EEXT, PTILEX)
 CALL DYADV(EEXT, PTILD, EXPTIL)
 DOTNE = DOTV(PRDIR, EEXT)
 DOTPN = DOTV(PTILD, PRDIR)
 TEMP1=SIGNORM(ALFA,PTILD, EEXT)
 CALL VDOTT(PTILD,ALFA,TMP311)
 CALL VDOTT(EEXT,ALFA, TMP312)
```

```
      CALL VDOTT(PTILD,ALFA,DUMV1)
 CALL DYADV(DUMV1, PRDIR, TMP331)
      CALL VDOTT(PTILD,ALFA,DUMV1)
 CALL DYADV(PRDIR, DUMV1, TMP332)
      CALL VDOTT(PTILD,ALFA,DUMV1)
      CALL VDOTT(EEXT,ALFA,DUMV2)
      CALL DYADV(DUMV1,DUMV2, TMP333)
      CALL DYADV(DUMV2,DUMV1, TMP334)
      DUMC1 = (D33/PMAX)*(DOTNE*DOTPN+TEMP1)
      CALL CTEN2(DUMC1,PRODNN,DUMT1)
      DUMC1 = (D33/PMAX)*DOTNE
      CALL TSUM(TMP331,TMP332,DUMT2)
      CALL CTEN2(DUMC1,DUMT2,DUMT2)
      CALL TSUM(DUMT1,DUMT2,DUMT1)
      DUMC1 = (D31/PMAX)*(TEMP1+DOTNE*DOTPN)
      CALL CTEN2(DUMC1,ALFA,DUMT2)
      DUMC1 = (D31/PMAX)*DOTNE
      CALL TSUM(TMP331,TMP332,DUMT3)
      CALL CTEN2(DUMC1,DUMT3,DUMT3)
      CALL TSUM(DUMT2,DUMT3,DUMT2)
      DUMC1 = -2.0*TEMP1
      CALL CTEN2(DUMC1,PRODNN,DUMT3)
      CALL CTEN2(DOTPN,TMP332,DUMT4)
      CALL TSUM(DUMT3,TMP333,DUMT3)
      CALL TSUM(DUMT3,TMP334,DUMT3)
      CALL TSUM(DUMT3,DUMT4,DUMT3)
      CALL TSUM(TMP331,TMP332,DUMT4)
      DUMC1 = -DOTNE
      CALL CTEN2(DUMC1,DUMT4,DUMT4)
      CALL TSUM(DUMT3,DUMT4,DUMT3)
      DUMC1 = D15/(2.0*PMAX)
      CALL CTEN2(DUMC1,DUMT3,DUMT3)
      CALL TSUM(DUMT1,DUMT2,DUMT1)
      CALL TSUM(DUMT1,DUMT3,AKIJPK)
C
c EPHT: epsilon_height (Eqn. 3.32)
c PHT: P_height (Eqn. 3.33)
C EPHT=EPTILD+AKIJPK
      CALL TSUM(EPTILD,AKIJPK,EPHT)
C PHT=PTILD+AIJPJ
      CALL VSUM(PTILD,AIJPJ,PHT)

C
C
C
C Calculate U_ij (3.20 and 4.8)
C
C
C
C Calculate U_ij (3.20 and 4.8)
c sigpr: sigma*n
```

```
c esigpr: E (sigma*n)
c sigpre: (sigma*n) E
c prsgpr: n(sigma*n)
c sgprpr: (sigma*n)n
c eextpr: En
c preext: nE
      CALL VDOTT(PRDIR,SIG,SIGPR)
 CALL DYADV(EEXT, SIGPR, ESIGPR)
 CALL DYADV(SIGPR, EEXT, SIGPRE)
 CALL DYADV(PRDIR, SIGPR, PRSGPR)
 CALL DYADV(SIGPR, PRDIR, SGPRPR)
 CALL DYADV(EEXT, PRDIR, EEXTPR)
 CALL DYADV(PRDIR, EEXT, PREEXT)
C UIJ=0.
      CALL INITT(UIJ,UIJ)
C NOTE THIS IS ONLY ASSIGNED GIVEN IF 1
      DUMC1 = 3.0
      CALL CTEN2(DUMC1,ESIGPR,DUMT1)
      DUMC1 = -5.0*DOTNE
      CALL CTEN2(DUMC1,PRSGPR,DUMT2)
      CALL TSUM(DUMT1,DUMT2,DUMT1)
      DUMC1 = -3.0*DOTNE
      CALL CTEN2(DUMC1,SGPRPR,DUMT2)
      CALL TSUM(DUMT1,DUMT2,DUMT1)
      DUMC1 = -3.0*SIGNN
      CALL CTEN2(DUMC1,EEXTPR,DUMT2)
      CALL TSUM(DUMT1,DUMT2,DUMT1)
      DUMC1 = -SIGNN
      CALL CTEN2(DUMC1,PREEXT,DUMT2)
      CALL TSUM(DUMT1,DUMT2,DUMT1)
      DUMC1 = 2.0*DOTNE
      CALL CTEN2(DUMC1,SIG,DUMT2)
      CALL TSUM(DUMT1,DUMT2,DUMT1)
      DUMC1 = -DOTNE*SIGNN
      CALL CTEN2(DUMC1,UNIT,DUMT2)
      CALL TSUM(DUMT1,DUMT2,DUMT1)
      DUMC1 = 8.0*DOTNE*SIGNN
      CALL CTEN2(DUMC1,PRODNN,DUMT2)
      CALL TSUM(DUMT1,DUMT2,DUMT1)
      DUMC1 = (D33/(PMAX*PRMAG)-(D31+D15)/(PMAX*PRMAG))
      CALL CTEN2(DUMC1,DUMT1,DUMT1)
 IF (PRMAG.GT.0.) CALL TEQT(UIJ,DUMT1,UIJ)
C
Ccccccccccccccccccccccccccccccccccccccc
ccALL FOR MECH ccccccccccccccccccccccccc
C
C
C Calculating H_ij, eqn.(5.7) & (7.21)
 EPSRNN = SIGNORM(EPSR, PRDIR, PRDIR)

c PSAT: defined in Landis (2004)
```

```
C PSAT= (PMAX-PMIN)/(EPT+EPC)*(EPSRNN+EPC)+PMIN


 PSAT= PMAX


 TEMP1 = H0E*PMAX**2/PSAT
 TEMP2 = 1-PRMAG/PSAT

C
 TEMP3 = H0E*PMAX*((PMAX-PMIN)/(EPT+EPC))**2
 & *(PRMAG/(PSAT*(PSAT-PRMAG)))**2
 & *(3.-2.*(PRMAG/PSAT))
C HIJ=(TEMP1/PSAT)*TEMP2**(-2)*PRODNN
      DUMC1 = (TEMP1/PSAT)*TEMP2**(-2)
      CALL CTEN2(DUMC1,PRODNN,DUMT1)
C & +TEMP1/(PSAT-PRMAG)*ALFA
      DUMC1 =TEMP1/(PSAT-PRMAG)
      CALL CTEN2(DUMC1,ALFA,DUMT2)
      CALL TSUM(DUMT1,DUMT2,HIJ)
C HIJKP: H_kij P_k, the last term in eqn. 3.28
C HIJKEP: (H_ikl)(EP_kl): the first term on the RHS of eqn. 3.29
C HIJKPEP: (H_kij)(P_k)EPHT_ij, see eqn. 3.31

C temp3 =0.
      TEMP3 = 0.
C hijkcof =0.
      HIJKCOF = 0.
C hijkpep=0.
      HIJKPEP = 0.
c EPRI1: Eqn. 6.6
 EPRI1=(1./3.)*TRACE(EPSR)

c EPRDEV: Eqn. 6.9
      CALL CTEN2(EPRI1,UNIT,DUMT1)
      CALL TSUM(EPSR,DUMT1,EPRDEV)
c EPRJ2: Eqn. 6.7
      CALL TDOTT(EPRDEV,EPRDEV,DUMT1)
      EPRJ2=SQRT((2./3.)*TRACE(DUMT1))

c EPRJ3: Eqn. 6.8

      CALL TDOTT(EPRDEV,EPRDEV,TMP331)
      CALL TDOTT(EPRDEV,TMP331,TMP332)
      TEMP1 = TRACE(TMP332)
      TEMP2 = (abs(TEMP1)*4./3.)**(1./3.)
      IF(TEMP1.lt.0.) TEMP2 = -TEMP2
      EPRJ3 = TEMP2
 IF(EPRJ2.NE.0.) go to 22
 HMEPEP=0.
 GO TO 30
```

```
 22 RATIOJ=EPRJ3/EPRJ2


C DF: first derivative of Fs
C DDF: second derivative of Fs


 IF(RATIOJ.GE.0.) GO TO 26
 F = -0.0965*RATIOJ**3+0.01*RATIOJ**6+0.8935
 DF = -0.2895*RATIOJ**2+0.06*RATIOJ**5
 DDF = -0.579*RATIOJ+0.3*RATIOJ**4
 GO TO 28


 26 F = -0.1075*RATIOJ**3-0.027*RATIOJ**6-0.028*RATIOJ**21+0.8935
 DF = -0.3225*RATIOJ**2-0.162*RATIOJ**5-0.588*RATIOJ**20
 DDF = -0.645*RATIOJ-0.81*RATIOJ**4-11.76*RATIOJ**19


C the first term in eqn. 3.31
 28 HMEPEP = HMEPEP2(F, DF, DDF, EPRJ2, EPRJ3, EPRDEV,
 & EPTILD,UNIT,EPC,EXPMM,H0M)


C The second term in the following hmepep comes from
C the interaction contribution
 30 HMEPEP=HMEPEP+TEMP3*SIGNORM(EPTILD, PRDIR, PRDIR)**2

      CALL CTEN2(TEMPONE,UIJ,DUMT1)
      CALL TSUM(HIJ,DUMT1,DUMT1)
 40 DENOM = HMEPEP+2.*HIJKPEP+SIGNORM(DUMT1, PTILD, PTILD)

      CALL TDOTT(EPHT,SIGINC,DUMT1)
 PLAST = (TRACE(DUMT1)+DOTV(PHT, EEXTIN))/DENOM
 42 CALL CTEN2(PLAST,EPTILD,EPSRIN)
      CALL CTEN1(PLAST,PTILD,PRIN)
C hijkp: the second term in eqn. 3.28
C hijkep: the first term in eqn. 3.29
      DUMC1 = HIJKCOF*DOTV(PRIN,PRDIR)
 46 CALL CTEN2(DUMC1,PRODNN,HIJKP)
      DUMC1 = HIJKCOF*SIGNORM(EPSRIN, PRDIR, PRDIR)
      CALL CTEN1(DUMC1,PRDIR,HIJKEP)
      CALL VDOTT(PRIN,HIJ,EBINC)

C sigbin =0.
      CALL INITT(SIGBIN,SIGBIN)
      IF (EPRJ2.NE.0.)
 & CALL HIJKLEP(F, DF, DDF, EPRJ2, EPRJ3, EPRDEV,
 & EPSRIN, UNIT,EPC,EXPMM,H0M,SIGBIN)
C SIGBIN=SIGBIN+HIJKP
```

```
      CALL TSUM(SIGBIN,HIJKP,SIGBIN)
C The second term in the follwoing expression comes from
C the interaction term
      DUMC1 = TEMP3*SIGNORM(EPSRIN,PRDIR,PRDIR)
      CALL CTEN2(DUMC1,PRODNN,DUMT1)
      CALL TSUM(SIGBIN,DUMT1,SIGBIN)
 60 CONTINUE

C       EB=EB+EBINC
      CALL VSUM(EB,EBINC,EB)
C       SIGB=SIGB+SIGBIN
      CALL TSUM(SIGB,SIGBIN,SIGB)
C       PR=PR+PRIN
      CALL VSUM(PR,PRIN,PR)
C       EPSR=EPSR+EPSRIN
      CALL TSUM(EPSR,EPSRIN,EPSR)
      CALL TDOTT(EPHT,SIGINC,DUMT1)
      DUMC1 = TRACE(DUMT1)/DENOM
      CALL CTEN2(DUMC1,EPHT,EPINS2)
      DUMC1 = DOTV(PHT,EEXTIN)/DENOM
      CALL CTEN2(DUMC1,EPHT,EPINE2)

      CALL TDOTT(EPHT,SIGINC,DUMT1)
      DUMC1 = TRACE(DUMT1)/DENOM
      CALL CTEN1(DUMC1,PHT,DINS2)
      DUMC1 = DOTV(PHT,EEXTIN)/DENOM
      CALL CTEN1(DUMC1,PHT,DINE2)
C1000       EPSINC= EPINS1+EPINS2+EPINE1+EPINE2
 1000 CALL TSUM(EPINS1,EPINS2,EPSINC)
      CALL TSUM(EPINE1,EPSINC,EPSINC)
      CALL TSUM(EPINE2,EPSINC,EPSINC)
C       DINC=DINS1+DINS2+DINE1+DINE2
      CALL VSUM(DINS1,DINS2,DINC)
      CALL VSUM(DINE1,DINC,DINC)
      CALL VSUM(DINE2,DINC,DINC)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCc
      END

Cokokokokokokokokokokokokokokokokokokokokokokokokokokokokokokokokok
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCMAIN FUNCTIONS CALLED
CC
 SUBROUTINE DEDOT(EEXTIN,PRMAG,PMAX,PRDIR,D33,D31,D15,EPINE1)
C       THIS SUBROUTINE CALCULATES (D_ijk)*(dot E)_k or EPINE1
      DOUBLE PRECISION PRO, NN(3,3), CTA, CTB, CTC, CTD, CTE, CTF,
 & TAA(3,3),TBB(3,3),TCC(3,3),TDD(3,3),TEE(3,3),
 & TFF(3,3)
      DOUBLE PRECISION EEXTIN(3), EPINE1(3,3)
      DOUBLE PRECISION UNIT(3,3),PRMAG,PMAX,PRDIR(3),
```

```
 & D33,D31,D15

CCCCCCCCCCCCCCC
C XFT = d E
C setup
      PRO = PRMAG/PMAX
 CALL DYADV(PRDIR,PRDIR,NN)
 CALL INITT(UNIT,UNIT)
 UNIT(1,1)=1.
 UNIT(2,2)=1.
 UNIT(3,3)=1.
C TermA
 CTA = PRO*D33*
 & (PRDIR(1)*EEXTIN(1)+
 & PRDIR(2)*EEXTIN(2)+
 & PRDIR(3)*EEXTIN(3))
 CALL CTEN2(CTA,NN,TAA)
C TermB
 CTB = PRO*D31*
 & (PRDIR(1)*EEXTIN(1)+
 & PRDIR(2)*EEXTIN(2)+
 & PRDIR(3)*EEXTIN(3))
 CALL CTEN2(CTB,UNIT,TBB)
 CALL TSUM(TAA,TBB,TBB)
C TermC
 CTC = -PRO*D31*
 & (PRDIR(1)*EEXTIN(1)+
 & PRDIR(2)*EEXTIN(2)+
 & PRDIR(3)*EEXTIN(3))
 CALL CTEN2(CTC,NN,TCC)
 CALL TSUM(TBB,TCC,TCC)
C TermD
 CTD = 0.5*PRO*D15
 CALL DYADV(PRDIR,EEXTIN,TDD)
 CALL CTEN2(CTD,TDD,TDD)
 CALL TSUM(TCC,TDD,TDD)
C TermE
 CTE = 0.5*PRO*D15
 CALL DYADV(EEXTIN,PRDIR,TEE)
 CALL CTEN2(CTD,TEE,TEE)
 CALL TSUM(TDD,TEE,TEE)
C TermF
 CTF = -D15*PRO*
 & (PRDIR(1)*EEXTIN(1)+
 & PRDIR(2)*EEXTIN(2)+
 & PRDIR(3)*EEXTIN(3))
 CALL CTEN2(CTF,NN,TFF)
 CALL TSUM(TEE,TFF,EPINE1)
 RETURN
 END
C
```

```
C
 SUBROUTINE PERMEDOT(EEXTIN, PERMI, DINE1)
C       THIS SUBROUTINE CALCULATES (K_ij)*(dot E)_j or DINE1

       DOUBLE PRECISION EEXTIN(3), DINE1(3)
       DOUBLE PRECISION PERMI,EPS11
       CALL CTEN1(PERMI,EEXTIN,DINE1)
 RETURN
 END
C
C
 SUBROUTINE DSIGDOT(SIGINC,PRDIR,PRMAG,PMAX,D33,D31,D15,DINS1)
C       THIS SUBROUTINE CALCULATES (D_ikl)*(dot SIGMA)_kl or DINS1
       DOUBLE PRECISION PRO, NT(3),VA(3),VB(3),VC(3),VD(3),VE(3),
 & CA, CB, CC, CD, CE
       DOUBLE PRECISION SIGINC(3,3), DINS1(3)
       DOUBLE PRECISION PRDIR(3),PRMAG,PMAX,D33,D31,D15

CCCCCCCCCCCCCC
C DF = d T
CC initalise setup
       PRO = PRMAG/PMAX
C TermA
       CALL VDOTT(PRDIR,SIGINC,NT)
       CA= D33*PRO*
 & (NT(1)*PRDIR(1)+
 & NT(2)*PRDIR(2)+
 & NT(3)*PRDIR(3))
       CALL CTEN1(CA,PRDIR,VA)
C TermB
       CB= -D31*PRO*
 & (SIGINC(1,1)+SIGINC(2,2)+SIGINC(3,3))
       CALL CTEN1(CB,PRDIR,VB)
       CALL VSUM(VA,VB,VB)
C TermC
       CC= -D31*PRO*
 & (NT(1)*PRDIR(1)+
 & NT(2)*PRDIR(2)+
 & NT(3)*PRDIR(3))
       CALL CTEN1(CC,PRDIR,VC)
       CALL VSUM(VB,VC,VC)
C TermD
       CD= D15*PRO
       CALL CTEN1(CD,NT,VD)
       CALL VSUM(VC,VD,VD)
C TermE
       CE= -D15*PRO*
 & (NT(1)*PRDIR(1)+
 & NT(2)*PRDIR(2)+
 & NT(3)*PRDIR(3))
       CALL CTEN1(CE,PRDIR,VE)
```

```
      CALL VSUM(VD,VE,DINS1)
 RETURN
 END
C
C
 SUBROUTINE SSIGDOT(SIGINC,POISS,YOUNGS, EPINS1)
C       THIS SUBROUTINE CALCULATES (S_iJkl)*(dot SIGMA)_kl or EPINS1

      DOUBLE PRECISION CXA, CXB, TA(3,3), TB(3,3)
      DOUBLE PRECISION SIGINC(3,3), EPINS1(3)
      DOUBLE PRECISION UNIT(3,3), POISS, YOUNGS
C steup
 CALL INITT(UNIT,UNIT)
 UNIT(1,1)=1.
 UNIT(2,2)=1.
 UNIT(3,3)=1.

C Term A
      CXA = ((1.+POISS)/YOUNGS)
      CALL CTEN2(CXA,SIGINC,TA)
C Term B
      CXB = (POISS/YOUNGS)*
 & (SIGINC(1,1)+SIGINC(2,2)+SIGINC(3,3))
      CALL CTEN2(CXB,UNIT,TB)
      CALL TSUM(TA,TB,EPINS1)
 RETURN
 END
C
Cokokokokokokokokokokokokokokokokokokokokokokokokokokokokokokokokokok
Cokokokokokokokokokokokokokokokokokokokokokokokokokokokokokokokokokok
      SUBROUTINE HIJKLEP(F, DF, DDF, EPRJ2, EPRJ3, EPRDEV,
 & DEP, UNIT,EPC,EXPMM,H0M, OUT)

C This subroutine calculates (H_ijkl)(epsilon_kl), Eqn. (6.3) & (6.14)

      DOUBLE PRECISION EPRDEV(3,3), DEP(3,3), HMLILJ(3,3),
 & ETAEPI(3,3),
 & UNIT(3,3), EQ614(3,3), OUT(3,3),
 & PART1(3,3), PART2(3,3), PART3(3,3), PART4(3,3),
cc
 & F,DF,DDF,EPRJ2,EPRJ3,
 & RATIOJ,EPBAR,TEMP1,TEMP2,
 & DUMC1,DUMC2,DUMC3,
 & DUMT1(3,3),DUMT2(3,3),DUMT3(3,3),DUMT4(3,3),
 & EPC,EXPMM,H0M,
 & fin1,fin2,fin3,fin4,fin5
      RATIOJ=EPRJ3/EPRJ2
      EPBAR = EPRJ2*F
      TEMP1 = EXPMM/((EPC-EPBAR)**2)
      fin1 = H0M
      fin2 = EPRJ2
```

```
      fin3 = EXPMM
      fin4 = EPBAR
      fin5 = EPC
      fin1 = EPBAR/EPC
      fin2 = (EXPMM/(1-EPBAR/EPC))
      fin3 = EXP(EXPMM/(1-EPBAR/EPC))
      fin4 = EPRJ2*EXP(EXPMM/(1-EPBAR/EPC))
      TEMP2= H0M*(EPRJ2*EXP(EXPMM/(1-EPBAR/EPC)))**2
C Eqn. 6.14, (L_i)(L_j)
      DUMC1 = (2./3.)*(F-RATIOJ*DF)/EPRJ2
      CALL CTEN2(DUMC1,EPRDEV,DUMT1)
      DUMC1 = (4.*DF)/(3.*EPRJ3**2)
      CALL TDOTT(EPRDEV,EPRDEV,DUMT2)
      CALL CTEN2(DUMC1,DUMT2,DUMT2)
      DUMC1 = (-0.5*EPRJ2**2)*DUMC1
      CALL CTEN2(DUMC1,UNIT,DUMT3)
      CALL TSUM(DUMT1,DUMT2,DUMT1)
      CALL TSUM(DUMT1,DUMT3,EQ614)

      DUMC1 = (2./(3.*EPC*EPRJ2**2))
      CALL CTEN2(DUMC1,EPRDEV,DUMT1)
      CALL CTEN2(TEMP1,EQ614,DUMT2)
      CALL TSUM(DUMT1,DUMT2,PART1)
      CALL TDOTT(PART1,DEP,DUMT1)
      DUMC1 = 2.*EPC*TEMP2*(DUMT1(1,1)+DUMT1(2,2)+DUMT1(3,3))
      CALL CTEN2(DUMC1,PART1,OUT)
      CALL TDOTT(EPRDEV,DEP,DUMT1)
      DUMC1 = (-4./(3.*EPRJ2**4))*
     & (DUMT1(1,1)+DUMT1(2,2)+DUMT1(3,3))
      CALL CTEN2(DUMC1,EPRDEV,DUMT1)
      DUMC1 = -(1./3.)*(DEP(1,1)+DEP(2,2)+DEP(3,3))
      CALL CTEN2(DUMC1,UNIT,DUMT2)
      CALL TSUM(DEP,DUMT2,DUMT2)
      DUMC1 = EPRJ2**(-2)
      CALL CTEN2(DUMC1,DUMT2,DUMT2)
      CALL TSUM(DUMT1,DUMT2,PART2)

C PART2
      DUMC1 = 2./(3.*EPC)
      CALL CTEN2(DUMC1,PART2,PART2)
C PART3
      CALL TDOTT(EQ614,DEP,DUMT1)
      DUMC1 = 2.*EXPMM*(EPC-EPBAR)**(-3)*
     & (DUMT1(1,1)+DUMT1(2,2)+DUMT1(3,3))
      CALL CTEN2(DUMC1,EQ614,PART3)
C ETAEPI=
      DUMC1 = ((4./9.)*(EPRJ2**3/EPRJ3**4)*DDF-(2./(9.*EPRJ2))*
     & (F-RATIOJ*DF)-(8./9.)*(EPRJ2**4/EPRJ3**5)*DF)
      DUMC1 = DUMC1 * (DEP(1,1)+DEP(2,2)+DEP(3,3))
      CALL CTEN2(DUMC1,UNIT,DUMT1)
C2
```

```
      DUMC1 = (2./(3.*EPRJ2)*(F-RATIOJ*DF))
      CALL CTEN2(DUMC1,DEP,DUMT2)
      CALL TSUM(DUMT1,DUMT2,DUMT1)
C3
      DUMC1 = (4.*DDF/(9.*EPRJ2*EPRJ3)-8.*DF/(9.*EPRJ3**2))
      CALL TDOTT(EPRDEV,DEP,DUMT2)
      DUMC2 = (DUMT2(1,1)+DUMT2(2,2)+DUMT2(3,3))*DUMC1
      CALL CTEN2(DUMC2,UNIT,DUMT2)
      DUMC2 = (DEP(1,1)+DEP(2,2)+DEP(3,3))*DUMC1
      CALL CTEN2(DUMC2,EPRDEV,DUMT3)
      CALL TSUM(DUMT2,DUMT3,DUMT2)
      CALL TSUM(DUMT1,DUMT2,DUMT1)
C4
      DUMC1 = (4.*DF)/(3.*EPRJ3**2)
      CALL TDOTT(DEP,EPRDEV,DUMT2)
      CALL TDOTT(EPRDEV,DEP,DUMT3)
      CALL TSUM(DUMT2,DUMT3,DUMT2)
      CALL CTEN2(DUMC1,DUMT2,DUMT2)
      CALL TSUM(DUMT1,DUMT2,DUMT1)
C5
      DUMC1 = (16.*EPRJ2**2*DF/
     & (9.*EPRJ3**5) - 8.*EPRJ2*DDF/(9.*EPRJ3**4))
      CALL TDOTT(EPRDEV,EPRDEV,DUMT2)
      CALL TDOTT(DUMT2,DEP,DUMT2)
      DUMC2 = (DUMT2(1,1)+DUMT2(2,2)+DUMT2(3,3))*DUMC1
      CALL CTEN2(DUMC2,UNIT,DUMT2)
      DUMC2 = (DEP(1,1)+DEP(2,2)+DEP(3,3))*DUMC1
      CALL TDOTT(EPRDEV,EPRDEV,DUMT3)
      CALL CTEN2(DUMC2,DUMT3,DUMT3)
      CALL TSUM(DUMT2,DUMT3,DUMT2)
      CALL TSUM(DUMT1,DUMT2,DUMT1)
C6
      DUMC1 = (4.*EPRJ3**2*DDF/(9.*EPRJ2**5)-
     & 4.*(F-RATIOJ*DF)/(9.*EPRJ2**3))
      CALL TDOTT(EPRDEV,DEP,DUMT2)
      DUMC1 = DUMC1*(DUMT2(1,1)+DUMT2(2,2)+DUMT2(3,3))
      CALL CTEN2(DUMC1,EPRDEV,DUMT2)
      CALL TSUM(DUMT1,DUMT2,DUMT1)

C7
      DUMC1 = (-8.*DDF/(9.*EPRJ2**3*EPRJ3))
      CALL TDOTT(EPRDEV,EPRDEV,DUMT2)
      CALL TDOTT(DUMT2,DEP,DUMT2)
      DUMC2 = DUMC1*(DUMT2(1,1)+DUMT2(2,2)+DUMT2(3,3))
      CALL CTEN2(DUMC2,EPRDEV,DUMT2)
      CALL TDOTT(EPRDEV,DEP,DUMT3)
      DUMC2 = DUMC1*(DUMT3(1,1)+DUMT3(2,2)+DUMT3(3,3))
      CALL TDOTT(EPRDEV,EPRDEV,DUMT3)
      CALL CTEN2(DUMC2,DUMT3,DUMT3)
      CALL TSUM(DUMT2,DUMT3,DUMT2)
      CALL TSUM(DUMT1,DUMT2,DUMT1)
```

```
C8
      DUMC1 = (16.*DDF/(9.*EPRJ2*EPRJ3**4)-32.*DF/(9.*EPRJ3**5))
      CALL TDOTT(EPRDEV,EPRDEV,DUMT2)
      CALL TDOTT(DUMT2,DEP,DUMT2)
      DUMC1 = DUMC1*(DUMT2(1,1)+DUMT2(2,2)+DUMT2(3,3))
      CALL TDOTT(EPRDEV,EPRDEV,DUMT2)
      CALL CTEN2(DUMC1,DUMT2,DUMT2)
      CALL TSUM(DUMT1,DUMT2,ETAEPI)
C PART4
      DUMC1 = TEMP1/(EPC-EPBAR)
      CALL CTEN2(DUMC1,ETAEPI,PART4)
C OUT=OUT+TEMP2*(PART2+PART3+PART4)
      CALL TSUM(PART2,PART3,DUMT1)
      CALL TSUM(DUMT1,PART4,DUMT1)
      CALL CTEN2(TEMP2,DUMT1,DUMT1)
      CALL TSUM(OUT,DUMT1,OUT)
 RETURN
 END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      FUNCTION HMEPEP2(F, DF, DDF, EPRJ2, EPRJ3, EPRDEV,
 & DEP,UNIT,EPC,EXPMM,H0M)

C This subroutine calculates (H_ijkl)(epsilon_kl), Eqn. (6.3) & (6.14)

      DOUBLE PRECISION EPRDEV(3,3), DEP(3,3), OUT(3,3),
 & DUMC1,
 & DUMT1(3,3),DUMT2(3,3),DUMT3(3,3),
 & F,DF,DDF,EPRJ2,EPRJ3,
 & UNIT(3,3),EPC,EXPMM,H0M


      CALL HIJKLEP(F, DF, DDF, EPRJ2, EPRJ3, EPRDEV,
 & DEP, UNIT,EPC,EXPMM,H0M, OUT)
      CALL TDOTT(OUT,DEP,DUMT1)
      HMEPEP2 = (DUMT1(1,1)+DUMT1(2,2)+DUMT1(3,3))
 RETURN
 END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC FUNCTIONS AND SUBROUTINES USED FOR THE MAIN SUBROUTINES CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C FUNCTIONSC
CCCCCCCCCCCCCCCC
C NORM = |PR|
C where PR is a vector
      FUNCTION NORM(PR)
      PARAMETER (limit=3)
      DOUBLE PRECISION P, PR(limit)
      P = PR(1)*PR(1)+PR(2)*PR(2)+PR(3)*PR(3)
      NORM = SQRT(P)
      END
```

```
C TRACE = TR(A)
C where A is a tensor of rank 1
      FUNCTION TRACE(A)
      PARAMETER (limit=3)
      DOUBLE PRECISION A(limit,limit)
      TRACE = A(1,1)+A(2,2)+A(3,3)
      END
C DOTV = A dot B
C where A and B are vectors
      FUNCTION DOTV(A,B)
      PARAMETER (limit=3)
      DOUBLE PRECISION A(limit), B(limit)
      DOTV = A(1)*B(1) + A(2)*B(2) + A(3)*B(3)
      END
C SIGNORM = C dot A dot B
C where A,C are vectors, B 3x3 and signorm is a variable
C       This subroutine calculates (B)(A)(B)
C       D=MATMUL(A,B)
      FUNCTION SIGNORM(A, B, C)
      DOUBLE PRECISION A(3,3), B(3), C(3), D(3)
      D(1) = A(1,1)*B(1)+A(1,2)*B(2)+A(1,3)*B(3)
      D(2) = A(2,1)*B(1)+A(2,2)*B(2)+A(2,3)*B(3)
      D(3) = A(3,1)*B(1)+A(3,2)*B(2)+A(3,3)*B(3)
      SIGNORM=DOTV(C,D)
 RETURN
 END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C SUBROUTINESC
CCCCCCCCCCCCCCCCCC
C CTEN1 = C * V
C where C is a tensor rank 0
C where V is a tesor rank 1
      SUBROUTINE CTEN1 (C,V,CV)
      PARAMETER (limit=3)
      DOUBLE PRECISION C, V(limit), CV(limit)
      CV(1) = C * V(1)
      CV(2) = C * V(2)
      CV(3) = C * V(3)
      END
C CTEN2 = C * T
C where C is a tensor rank 0
C where T is a tesor rank 2
      SUBROUTINE CTEN2 (C,T,CT)
      PARAMETER (limit=3)
      DOUBLE PRECISION C, T(limit,limit), CT(limit,limit)
      CT(1,1) = C * T(1,1)
      CT(1,2) = C * T(1,2)
      CT(1,3) = C * T(1,3)
      CT(2,1) = C * T(2,1)
      CT(2,2) = C * T(2,2)
      CT(2,3) = C * T(2,3)
```

```
      CT(3,1) = C * T(3,1)
      CT(3,2) = C * T(3,2)
      CT(3,3) = C * T(3,3)
      END
C VDOTT = V dot T
C where V is a tensor rank 1
C where T is a tensor rank 2
      SUBROUTINE VDOTT (V,T,VEC)
      PARAMETER (limit=3)
      DOUBLE PRECISION V(limit),T(limit,limit), VEC(limit)
      VEC(1) = V(1)*T(1,1)+V(1)*T(1,2)+V(1)*T(1,3)
      VEC(2) = V(2)*T(2,1)+V(2)*T(2,2)+V(2)*T(2,3)
      VEC(3) = V(3)*T(3,1)+V(3)*T(3,2)+V(3)*T(3,3)
      END
C TDOTT = T dot T
C where A is a tensor rank 2
C where B is a tensor rank 2
C where C is a tensor rank 2
      SUBROUTINE TDOTT (A,B,C)
      DOUBLE PRECISION A(3,3),B(3,3),C(3,3)
      C(1,1) = A(1,1)*B(1,1)+A(1,2)*B(2,1)+A(1,3)*B(3,1)
      C(1,2) = A(1,1)*B(1,2)+A(1,2)*B(2,2)+A(1,3)*B(3,2)
      C(1,3) = A(1,1)*B(1,3)+A(1,2)*B(2,3)+A(1,3)*B(3,3)
      C(2,1) = A(2,1)*B(1,1)+A(2,2)*B(2,1)+A(2,3)*B(3,1)
      C(2,2) = A(2,1)*B(1,2)+A(2,2)*B(2,2)+A(2,3)*B(3,2)
      C(2,3) = A(2,1)*B(1,3)+A(2,2)*B(2,3)+A(2,3)*B(3,3)
      C(3,1) = A(3,1)*B(1,1)+A(3,2)*B(2,1)+A(3,3)*B(3,1)
      C(3,2) = A(3,1)*B(1,2)+A(3,2)*B(2,2)+A(3,3)*B(3,2)
      C(3,3) = A(3,1)*B(1,3)+A(3,2)*B(2,3)+A(3,3)*B(3,3)
      END
C VSUM = V + V
C where V is a tensor rank 1
C where V is a tensor rank 1
      SUBROUTINE VSUM (A,B,C)
      PARAMETER (limit=3)
      DOUBLE PRECISION A(limit),B(limit), C(limit)
      C(1) = A(1)+B(1)
      C(2) = A(2)+B(2)
      C(3) = A(3)+B(3)
      END
C TSUM = T + T
C where T is a tensor rank 2
C where T is a tensor rank 2
      SUBROUTINE TSUM (A,B,C)
      PARAMETER (limit=3)
      DOUBLE PRECISION A(limit,limit),B(limit,limit), C(limit,limit)
      C(1,1) = A(1,1)+B(1,1)
      C(1,2) = A(1,2)+B(1,2)
      C(1,3) = A(1,3)+B(1,3)
      C(2,1) = A(2,1)+B(2,1)
      C(2,2) = A(2,2)+B(2,2)
```

```
      C(2,3) = A(2,3)+B(2,3)
      C(3,1) = A(3,1)+B(3,1)
      C(3,2) = A(3,2)+B(3,2)
      C(3,3) = A(3,3)+B(3,3)
      END
C DYADV = AB
C where A is a tensor rank 1
C where B is a tensor rank 1
      SUBROUTINE DYADV (A,B,C)
      PARAMETER (limit=3)
      DOUBLE PRECISION A(limit),B(limit), C(limit,limit)
      C(1,1) = A(1)*B(1)
      C(1,2) = A(1)*B(2)
      C(1,3) = A(1)*B(3)
      C(2,1) = A(2)*B(1)
      C(2,2) = A(2)*B(2)
      C(2,3) = A(2)*B(3)
      C(3,1) = A(3)*B(1)
      C(3,2) = A(3)*B(2)
      C(3,3) = A(3)*B(3)
      END
C INITV = V = 0,0
C where V is a tensor rank 1
      SUBROUTINE INITV (A,B)
      PARAMETER (limit=3)
      DOUBLE PRECISION A(limit),B(limit)
            B(1) = 0.0
            B(2) = 0.0
            B(3) = 0.0
      END
C INITT = T = 0.0
C where T is a tensor rank 2
      SUBROUTINE INITT (A,B)
      PARAMETER (limit=3)
      DOUBLE PRECISION A(limit,limit),B(limit,limit)
            B(1,1) = 0.0
            B(1,2) = 0.0
            B(1,3) = 0.0
            B(2,1) = 0.0
            B(2,2) = 0.0
            B(2,3) = 0.0
            B(3,1) = 0.0
            B(3,2) = 0.0
            B(3,3) = 0.0
      END
C VEQV = A=B
C where A&V are tensors rank 1
      SUBROUTINE VEQV (A,B,C)
      PARAMETER (limit=3)
      DOUBLE PRECISION A(limit),B(limit),C(limit)
            C(1) = B(1)
```

```
            C(2) = B(2)
            C(3) = B(3)
        END
C TEQT = A=B
C where A&B are tensors rank 2
        SUBROUTINE TEQT (A,B,C)
        DOUBLE PRECISION A(3,3),B(3,3),C(3,3)
            C(1,1) = B(1,1)
            C(1,2) = B(1,2)
            C(1,3) = B(1,3)
            C(2,1) = B(2,1)
            C(2,2) = B(2,2)
            C(2,3) = B(2,3)
            C(3,1) = B(3,1)
            C(3,2) = B(3,2)
            C(3,3) = B(3,3)
        END
```

A.2 Mclan.C

Mclan.C:

```
#include <iostream>
#include <math.h>
#include <stdlib.h>
#include "file_io.h"
#include "data_container.h"
#include "keyword.h"
#include "physlib.h"
#include "material.h"
#include "material_model_enum.h"
#include "material_model.h"
#include "material_data.h"
#include "region.h"
#include "token_stream.h"
#include "mclan.h"
using namespace std;
#ifdef SNL_QSEM
//THIS INITIALIZES THE EXTERNAL FORTRAN SUBROUTINE
extern "C" {
 void FORTRAN(nonlin)(
       double*,
       double*,
       double*,
       double*,
       double*,
       double*,
       double*,
       double*,
       double*,
       double* ui,
       double*,
       double*);
}
// THE FOLLOWING ARE TO BE SET BY USER IN INPUT FILE
const char *Mclan::ParamNames[Mclan::MAX_PARAM] =
 {
 /** Coersive Field */
 "E0",

 /** Polarization coresponding to Coersive Field */
 "P0",

 /** Strain coresponding to Coersive Field */
 "S0",

 /** Piezoelectric constants */
 "D33",
 "D31",
```

```
 "D15",

 /** Dielectric Permittivity */
 "K",

 /** Youngs Modulus */
 "Y",

 /** Poissons Ratio */
 "V",
 };
Mclan::Mclan() : Material_Model() { }
Mclan::Mclan(int i) : Material_Model(i, Mclan::MAX_PARAM) { }
Mclan::Mclan(const Mclan& src) : Material_Model(src)
{
 Copy(src);
}
void Mclan::Copy(const Material_Model& mmsrc)
{
 Material_Model::Copy(mmsrc);
}
Mclan::~Mclan() {}
/*****************************************************************************/
Token Mclan::Parse_Model_Parameters(Token_Stream *ts)
/*****************************************************************************/
{
 Keyword parameter_table[] = {

 {"E0", E0, Get_Real_Token},
 {"P0", P0, Get_Real_Token},
 {"S0", S0, Get_Real_Token},
 {"D33", D33, Get_Real_Token},
 {"D31", D31, Get_Real_Token},
 {"D15", D15, Get_Real_Token},
 {"K", K, Get_Real_Token},
 {"Y", Y, Get_Real_Token},
 {"V", V, Get_Real_Token},
 };
 int N = sizeof(parameter_table)/sizeof(Keyword);
 Token token = Material_Model::Parse_Model_Parameters(ts,parameter_table,N);
 return token;
}
/****************111
****************SET UP
****************/
/*************************************/
ErrorStatus Mclan::Set_Up(Material *mat, Region *region)
/*****************************************************************************/
{
      ///AAA
 // register element variables
```

```
Variable_Attributes *attr;

attr = region->getDataRegister().Find_Variable("ELECTRIC_FIELD");
if (!attr || attr->Centering() != ELEMENT_VAR || attr->Type()!=VECTOR_VAR)
code_error(TASK_0_INFORMATION, "MCLAN::Set_Up", "ELECTRIC_FIELD not defined");
else
EFIELD = attr->Index();
     ///BBB
// register material model variables
STRESS = mat->Register_Variable(SYMTENSOR_VAR,"STRESS", "MCLAN",
MMODE_IOPUT, PRESSURE_UNITS, NO_REMAP);
STRAIN = mat->Register_Variable(SYMTENSOR_VAR,"STRAIN", "MCLAN",
MMODE_IOPUT, NONDIMENSIONAL_UNITS, NO_REMAP);
ROTATION = mat->Register_Variable(TENSOR_VAR,"ROTATION", "MCLAN",
MMODE_INPUT, NONDIMENSIONAL_UNITS, NO_REMAP);
STRETCH = mat->Register_Variable(SYMTENSOR_VAR,"STRETCH", "MCLAN",
MMODE_INPUT, NONDIMENSIONAL_UNITS, NO_REMAP);
MPOLARIZATION = mat->Register_Variable(VECTOR_VAR,"MECHANICAL_POLARIZATION","MCLAN",
MMODE_IOPUT, NONDIMENSIONAL_UNITS,NO_REMAP);
//used to store previous Efield
EF_PRE = mat->Register_Variable(VECTOR_VAR,"EF_PRE","MCLAN",
MMODE_IOPUT, NONDIMENSIONAL_UNITS,NO_REMAP);
//REMNANT TERMS
//used to store the remnant porlarization
PR = mat->Register_Variable(VECTOR_VAR,"PR","MCLAN",
MMODE_IOPUT, NONDIMENSIONAL_UNITS,NO_REMAP);
//used to store the remnant Electric Field
EB = mat->Register_Variable(VECTOR_VAR,"EB","MCLAN",
MMODE_IOPUT, NONDIMENSIONAL_UNITS,NO_REMAP);
//used to store the remnant Electric Field
SIGB = mat->Register_Variable(TENSOR_VAR,"SIGB","MCLAN",
MMODE_IOPUT, NONDIMENSIONAL_UNITS,NO_REMAP);
//used to store the remnant Electric Field
EPSR = mat->Register_Variable(TENSOR_VAR,"EPSR","MCLAN",
MMODE_IOPUT, NONDIMENSIONAL_UNITS,NO_REMAP);
//
ELECTRIC_DISP = mat->Register_Variable(VECTOR_VAR,"ELECTRIC_DISPLACEMENT","MCLAN",
MMODE_IOPUT, NONDIMENSIONAL_UNITS,NO_REMAP);
PERMITTIVITY = mat->Register_Variable(SYMTENSOR_VAR,"PERMITTIVITY","MCLAN",
MMODE_IOPUT,NONDIMENSIONAL_UNITS,NO_REMAP);
FE_POLARIZATION =mat->Register_Variable(VECTOR_VAR,"FE_POLARIZATION","MCLAN",
MMODE_IOPUT,NONDIMENSIONAL_UNITS,NO_REMAP);
FE_POL_INIT =mat->Register_Variable(VECTOR_VAR,"FE_POL_INIT","MCLAN",
MMODE_IOPUT,NONDIMENSIONAL_UNITS,NO_REMAP);

MAX_POL_MAG = mat->Register_Variable(SCALAR_VAR,"MAX_POL_MAG","SC_PZT",
MMODE_IOPUT, NONDIMENSIONAL_UNITS, NO_REMAP);
      ///CCC
//register obligatory material variables
/*** NOTE: these are emd stuff so keep them ***/
BULK_MODULUS = mat->Register_Variable(SCALAR_VAR,"BULK_MODULUS", "MCLAN",
```

```
 MMODE_OUTPUT, PRESSURE_UNITS, VOLUME_REMAP);
 SHEAR_MODULUS = mat->Register_Variable(SCALAR_VAR,"SHEAR_MODULUS", "MCLAN",
 MMODE_OUTPUT, PRESSURE_UNITS, VOLUME_REMAP);
 for(int i=0;i<NFECV;i++)
 FE_ELASTIC_CONSTS[i] = FE_ELASTIC_CONSTS_OFFSET + i*NUM_ELASTIC_CONSTS;
 for(int i=0;i<NFECV;i++)
 FE_PIEZOELECTRIC_CONSTS[i] = i*NUM_PIEZOELECTRIC_CONSTS;
 for(int i=0;i<NFECV;i++)
 FE_PERMITTIVITY_CONSTS[i] = i*NUM_PERMITTIVITY_CONSTS;
 status = SET;
 return 0;
 }
 /*********************222
 **********************INITIALIZE STATE
 **********************/
 /*****************************************************************************/
 ErrorStatus Mclan::Initialize_State(Data_Container*,
  Material_Data* var,
  External_Material_Data* emd)
 /*****************************************************************************/
  {
 ///AAA
 ///THIS PART IS JUST USED ONE TIME TO INITIALSE the Setup
  if(emd->Status() != External_Material_Data::EMD_INITIALIZED) {
  /** for the first element, allocate the emd storage */
  int emd_data_len = TOTAL_EMD_DATA;              /*defined in .h(#of derived constants) */
  int emd_vector_len = NFECV; // polarztn vector storage */
  int emd_tensor_len = TOTAL_TENSOR_DATA; /*< the material basis */
  emd->Allocate(emd_data_len, emd_vector_len, emd_tensor_len);
 //THIS MUST BE CALLED AS ALEGRA CRASHES WITHOUT A PERMITTIVITY
  Compute_Lab_Constants(emd);
  emd->Status(External_Material_Data::EMD_INITIALIZED);
 //OK NOW IT IS INITALIZED
  }
 ///BBB
 /******************/
 /* HERE THE TENSORS WHICH WILL BE USED IN EACH INCREMENTAL STEP ARE SET
INITIALY TO 0 */
  Tensor ATzero(0.0, 0.0, 0.0,
  0.0, 0.0, 0.0,
  0.0, 0.0, 0.0);

  Vector AVzero(0.0, 0.0, 0.0);

  var->Vector_Var(PR) = AVzero;
  var->Vector_Var(EB) = AVzero;
  var->Tensor_Var(SIGB) = ATzero;
  var->Tensor_Var(EPSR) = ATzero;
  /** compute the fe permittivity this relates to the call above, and is required so that alegra has a
permitivity and does not crash at this point */
  Real *cp;
```

```
 Real *p = (double *) &(var->SymTensor_Var(PERMITTIVITY));
 for(int i=0;i<6;i++) p[i] = 0.0;
 for(int i=0;i<NFECV;i++) {
 cp = &(emd->Data()[FE_PERMITTIVITY_CONSTS[i]]);
 p[0] += cp[0];p[3] += cp[1];p[5] += cp[2];
 p[1] += cp[4];p[4] += cp[5];
 p[2] += cp[8];
 }

 /* every element needs a shear modulus and bulk modulus for timestep
 calculations */
 var->Scalar_Var(BULK_MODULUS) = param[Y]/(3*(1-2*param[V]));
 var->Scalar_Var(SHEAR_MODULUS) = param[Y]*(1/(2*(1-param[V])));

 return 0;
}
/***************333
***************UPDATE STATE:
      this is where the model goes
***************/
//*********************************************
ErrorStatus Mclan::Update_State(Data_Container* topo_vars,
 Real delt,
 int,
 Material* mat,
 Data_Container* var_old,
 Data_Container* var_new,
 External_Material_Data* emd)
//*****************************************************************************
{
////////////////////////////////////
 //inputs for nonlin.F model//
////////////////////////////////////
/* The following are all retrieved from the stored data at the begining of each successive step */
// Remnant Polarization
 Vector Prem              = var_old->Vector_Var(PR);
//Back Electric Field
 Vector Eback             = var_old->Vector_Var(EB);
//Back Stress
 Tensor SIGback       = var_old->Tensor_Var(SIGB);
//Remanant Strain
 Tensor EPSrem            = var_old->Tensor_Var(EPSR);
// identity matrix
 Tensor id( 1.0, 0.0, 0.0,
 0.0, 1.0, 0.0,
 0.0, 0.0, 1.0);
/////////
/*Current Electric field, Alegra creates this from the current mesh configuration */
 Vector ce_field        = topo_vars->Vector_Var(EFIELD);
/* this is the value of the electric field from the previous time step */
 Vector oe_field       = var_old->Vector_Var(EF_PRE);
```

```
/*now EF the incremental change in Efeild can be derived */
 Vector EF       = ce_field - oe_field;
 Real e_field_mag       = Norm(EF);
/////////
//STRESS
 Tensor TF(0.0, 0.0, 0.0,
 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0);
////////
//ELECTRIC DISPLACMENT
 Vector d_field        = var_old->Vector_Var(ELECTRIC_DISP);
 Vector DF(0.0, 0.0, 0.0);
///////
//STRAIN
//XF = increment of strain
 Tensor XF(0.0, 0.0, 0.0,
 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0);
 SymTensor x_field = var_old->SymTensor_Var(STRAIN);
 Tensor ox_field = x_field;
///////
//INPUT CONSTANTS
/* Tensor ceos(       param[EPS11], param[D33], param[D31],
              param[D15], param[PFRAC], param[V],
              param[YM], param[CFIELD], 0.0);
*/
{
 FORTRAN(nonlin)(
       /* input */
       (double*) &ce_field,
       (double*) &oe_field,
       (double*) &EF,
       (double*) &TF,
       (double*) &TF,
       /*model spacifics */
       (double*) &Prem,
       (double*) &Eback,
       (double*) &EPSrem,
       (double*) &SIGback,
       /* user specified input */
       param,
       /* output */
       (double*) &DF,
       (double*) &XF);
}
/* Now the increment of Electric Displacment can be added to the previous value */
 var_new->Vector_Var(ELECTRIC_DISP) = d_field + DF;
/* the new value of strain as a tensor */
 XF = ox_field + XF;

/* storage requires this to be changed into the form of a symTensor */
```

```
 SymTensor XFS(XF.XX(), XF.XY(), XF.XZ(),
 XF.YY(), XF.YZ(),
 XF.ZZ());
 var_new->SymTensor_Var(STRAIN) = XFS;
//The following must simply be stored for the following itteration
 var_new->Vector_Var(PR) = Prem;
 var_new->Vector_Var(EB) = Eback;
 var_new->Tensor_Var(SIGB) = SIGback;
 var_new->Tensor_Var(EPSR) = EPSrem;
//at the end the efild must be stored away as a previous value..
 var_new->Vector_Var(EF_PRE) = ce_field;
 return 0;
}
/************555
*************COMPUTE LAB CONSTANTS
      this is where we go from to some/frame lab/frame
*************/
/*****************************************************************************/
ErrorStatus Mclan::Compute_Lab_Constants(External_Material_Data* emd)
/*****************************************************************************/
{

/* THIS IS SIMPLY AN ALEGRA REQUIREMENT SO IT STANDS AS SUCH */
 /** compute permittivity */
 {
 Tensor perm(param[K],0.0,0.0,
 0.0,param[K],0.0,
 0.0,0.0,param[K]);
 Real *permlb, *prm;
 for(int M=0; M<NFECV; M++) {
 permlb = &(emd->Data()[FE_PERMITTIVITY_CONSTS[M]]);
 Tensor permlbt = perm;
 prm = (double*)&permlbt;
 for(int i=0;i<9;i++) permlb[i] = prm[i];
 }
 }

 return 0;

}
#endif
```

A.3 Mclan.h

This is the header file implamented in association with the mclan.C main file.

Mclan.h:

```
#ifndef mclanH
#define mclanH
#include "code_types.h"
#include "material_data.h"
#ifdef SNL_QSEM
class Mclan : public Material_Model
{
 public:
 enum ParamType
 {
 /** Coersive Field */
 E0,

 /** Polarization coresponding to Coersive Field */
 P0,

 /** Strain coresponding to Coersive Field */
 S0,

 /** Piezoelectric constants */
 D33,
 D31,
 D15,

 /** Dielectric Permittivity */
 K,

 /** Youngs Modulus */
 Y,
 /** Poissons Ratio */
 V,


 MAX_PARAM
 };
 static const char* ParamNames[MAX_PARAM];
 Mclan();
 Mclan(int);
 Mclan(const Mclan&);
 ~Mclan();
 int Tag() const { return Material_Model::Tag("Mclan"); }
 MM_Type Type() const { return MMT_UNTYPED;}
 const char *Name() const { return "MC LAN"; }
 int Num_Params() const { return MAX_PARAM; }
 char** Get_Parameter_Names() const {return (char**) ParamNames;}
 virtual Token Parse_Model_Parameters(Token_Stream *);
 virtual bool Model_Supports_Two_State() {return true;}
```

```
ErrorStatus Compute_Lab_Constants(External_Material_Data* emd);

ErrorStatus TensorToArray(Real C[3][3][3][3], Real *c);
ErrorStatus TensorToArray(Real C[3][3][3], Real *c);


ErrorStatus Set_Up(Material*, Region*);

ErrorStatus Initialize_State(Data_Container*,
Material_Data*,
External_Material_Data*);

ErrorStatus Update_State(Data_Container* topo_vars,
Real delt,
int geometry,
Material* mat,
Data_Container* var,
Data_Container* /*var_new*/,
External_Material_Data* emd);
private:
// // constants for this model
enum { NFECV = 1, // the following are used for storage
NUM_DERIVED_CONSTS = 47,
NUM_PIEZOELECTRIC_CONSTS = 18,
NUM_PERMITTIVITY_CONSTS = 9,
NUM_ELASTIC_CONSTS = 36 };
enum EMD_Data_Offset {DERIVED_CONSTS_OFFSET = 0,
                //note always add the previous&the contents of the previous
FE_ELASTIC_CONSTS_OFFSET = NUM_DERIVED_CONSTS,

                FE_PIEZOELECTRIC_CONSTS_OFFSET = FE_ELASTIC_CONSTS_OFFSET +
                                NFECV*NUM_ELASTIC_CONSTS,
                FE_PERMITTIVITY_CONSTS_OFFSET = FE_PIEZOELECTRIC_CONSTS_OFFSET
+
                                NFECV*NUM_PIEZOELECTRIC_CONSTS,
                TOTAL_EMD_DATA = FE_PERMITTIVITY_CONSTS_OFFSET +
                        NFECV*NUM_PERMITTIVITY_CONSTS
                };
enum EMD_Vector_Offset {BINS = 0, FEBINS = 0};
enum EMD_Tensor_Offset {MATERIAL_BASIS=0,
FEROT = 1,
TOTAL_TENSOR_DATA = FEROT+NFECV};
// variable ids
// input:
Element_Data_Variable DEFRATE;
Element_Data_Variable EFIELD;
Element_Data_Variable STRETCH;
Element_Data_Variable ROTATION;
// ioput:
Material_Data_Variable STRESS;
Material_Data_Variable STRAIN;
```

```
 Material_Data_Variable EF_PRE;
 Material_Data_Variable PR;
 Material_Data_Variable EB;
 Material_Data_Variable SIGB;
 Material_Data_Variable EPSR;
 Material_Data_Variable ELECTRIC_DISP;
 Material_Data_Variable MPOLARIZATION;
 Material_Data_Variable PERMITTIVITY;
 // output:
 Material_Data_Variable FE_POLARIZATION;
 Material_Data_Variable FE_POL_INIT;
 Material_Data_Variable MAX_POL_MAG;
 Material_Data_Variable SPEC_VOL;
 // State
 Material_Data_Variable FEBIN;
 Material_Data_Variable FEBIN0;
 //Obligatory
 Material_Data_Variable BULK_MODULUS;
 Material_Data_Variable SHEAR_MODULUS;
 Material_Data_Variable SPECIFIC_HEAT_VOL;

 //material model constants offsets
 int FE_ELASTIC_CONSTS[NFECV];
 int FE_PIEZOELECTRIC_CONSTS[NFECV];
 int FE_PERMITTIVITY_CONSTS[NFECV];
 void Copy(const Material_Model&);
};
#endif
#endif
```

Appendix B. INPUT OPTIONS

B.1 Sample Input

This is a sample input file for applying a cyclical potential difference ranging from +-1.0 E6 (V/m) to a sample of 2x2x2cm. The material has its values set to that of PZT. There are periodic boundry conditions applied to the sides without the applied potiential difference.

File (mclan.inp):

```
$————————————————BEGIN_QA————————————————-
$ ID: Mclan.inp
$ Title: ?
$ Category: ?
$ Physics: qsem_physics
$ Dimension: 3D
$ Owner:  Morgan Allen Brown
$
$ Description:
$
$ ?
$
$ References: ?
$ Directory: ?
$ Tags: ?
$ CVS: ?
$————————————————END_QA————————————————
Units, SI
Termination cycle 4700
Emit plot, cycle interval=25
Emit output, cycle interval=25
Plot variable
 no underscores
 ELECTRIC POTENTIAL, as, epot
 ELECTRIC DISPLACEMENT, avg, as, edisp
 EF PRE, avg, as, efpre
 ELECTRIC FIELD, as, efield
 STRAIN, avg, as, strain
 PR, avg, as, PR
 PERMITTIVITY, avg, as, perm
end
quasistatic electric
 external e, x 0. y 0. z 1.

 start = 0.
 stop = 1.25
 time step = 0.0003
 circuit node 1 sideset 6 fixedv 0.0 $ min x face
 circuit node 2 sideset 5 functionv 7$ max x face
```

```
 function 7
0.0 0. $time applied-voltage
.05 4320000.
.10 8640000.
.15 12960000.
.20 17280000.
.25 21600000.

.30 17280000.
.35 12960000.
.40 8640000.
.45 4320000.
.50 0.

.55 -4320000.
.60 -8640000.
.65 -12960000.
.70 -17280000.
.75 -21600000.

.80 -17280000.
.85 -12960000.
.90 -8640000.
.95 -4320000.
1.00 0.

1.05 4320000.
1.10 8640000.
1.15 12960000.
1.20 17280000.
1.25 21600000.
 end
 region
 periodic bc, nodeset 1, translate, x 20. y 0. z 0., nodeset 2
 periodic bc, nodeset 3, translate, x 0. y 20. z 0., nodeset 4
$
 periodic bc, nodeset 11, translate, x 20. y 20. z 0., nodeset 12
 periodic bc, nodeset 13, translate, x 20. y -20. z 0., nodeset 14
 block 2
 material 1
 end
$ block 1
$ material 2
$ end
 end
end
aztec
 solver, cg
 scaling, sym_diag
 conv norm, anorm
 precond, ls
```

```
 poly ord, 9
 output, none
 tol= 1.e-12
end
$$$$$$$$$$$$$$$$$$$$$$$$$ material models $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
END
MATERIAL 2
 model 1
END
model 2 mclan
 E0 = 0.35e6
 P0 = 0.3133
 S0 = 2.75e7
 D33 = 6.0e-10
 D31 = -3.0e-10
 D15 = 9.0e-10
 K = 3.0e-8
 Y = 7.0e10
 V = 0.4

end
exit
```

B.2 Cubit Input

This is a series of cubit journal files which can be used by Sandia National Laboratories Cubit program to recreate a genesis meshs. The gemometry is that of a 2x2x2cm setup. There is a sideset placed in the z direction for the applicaiton of potential difference and periodic boundry conditions set on the remaining surfaces and curves. The first, Cube.jou has no porosity for initial validation, Sphere.jou has a center sphere and Cylinder.jou and Cylinder_Rotated.jou create center cylinders with the mesh and applied sideses for both cylinder curvatures.

File (Cube.jou):

```
brick x 20
volume 1 size .2
volume 1 scheme auto
mesh volume 1
block 2 volume all
sideset 5 surface 2
sideset 6 surface 1
x
nodeset 1 surface 4
nodeset 2 surface 6
y
nodeset 3 surface 3
nodeset 4 surface 5

xy
nodeset 11 curve 9
nodeset 12 curve 12
x-y
nodeset 13 curve 11
nodeset 14 curve 10
export genesis "mclan.gen"
```

File (Sphere.jou):

```
reset
brick x 10
sphere radius 5
body 2 move x 5 y -5 z 5
chop body 1 with 2
$$$$$$$
body all copy reflect x
volume 5 to 6 move x 10
body all copy reflect z
volume 7 to 10 move z 10
body all copy reflect y
volume 11 to 18 move y -10
$$$$$$$
imprint all
merge all
volume all size .5
curve 22 scheme bias .9 start vertex 6
```

```
curve 23 scheme bias .9 start vertex 2
curve 24 scheme bias .9 start vertex 4
curve 45 scheme bias .9 start vertex 29
curve 55 scheme bias .9 start vertex 37
curve 100 scheme bias .9 start vertex 67
vol 4 scheme sweep source surface 5 4 2 target surface 8
mesh vol 4
volume 4 smooth scheme laplacian
smooth volume 4
volume 4 smooth scheme mean ratio
smooth volume 4
vol 10 scheme sweep source surface 44 45 46 target surface 38
mesh vol 10
volume 10 smooth scheme laplacian
smooth volume 10
volume 10 smooth scheme mean ratio
smooth volume 10
vol 6 scheme sweep source surface 22 23 24 target surface 16
mesh vol 6
volume 6 smooth scheme laplacian
smooth volume 6
volume 6 smooth scheme mean ratio
smooth volume 6
vol 8 scheme sweep source surface 33 34 35 target surface 27
mesh vol 8
volume 8 smooth scheme laplacian
smooth volume 8
volume 8 smooth scheme mean ratio
smooth volume 8
vol 12 scheme sweep source surface 55 56 57 target surface 49
mesh vol 12
volume 12 smooth scheme laplacian
smooth volume 12
volume 12 smooth scheme mean ratio
smooth volume 12
vol 14 scheme sweep source surface 66 67 68 target surface 60
mesh vol 14
volume 14 smooth scheme laplacian
smooth volume 14
volume 14 smooth scheme mean ratio
smooth volume 14
vol 16 scheme sweep source surface 77 78 79 target surface 71
mesh vol 16
volume 16 smooth scheme laplacian
smooth volume 16
volume 16 smooth scheme mean ratio
smooth volume 16
vol 18 scheme sweep source surface 88 89 90 target surface 82
mesh vol 18
volume 18 smooth scheme laplacian
smooth volume 18
```

```
volume 18 smooth scheme mean ratio
smooth volume 18
volume 3 5 7 9 11 13 15 17 scheme tetprimitive
mesh vol 3 5 7 9 11 13 15 17
block 2 vol 2 4 6 8 10 12 14 16 18
block 1 vol 3 5 7 9 11 13 15 17
nodeset 1 surface 4
nodeset 2 surface 6
nodeset 3 surface 3
nodeset 4 surface 5
nodeset 11 curve 9
nodeset 12 curve 12
nodeset 13 curve 11
nodeset 14 curve 10
sideset 5 surface 2 24 57 68
sideset 6 surface 35 46 79 90
export genesis "mclan.gen"
```

File (Cylinder.jou):

```
brick x 6.5 y 20 z 20
create Cylinder height 6.5 radius 5
rotate volume 2 about Y angle 90
subtract vol 2 from vol 1
create Cylinder height 6.5 radius 5
$rotate volume 2 about Y angle 90
rotate volume 3 about Y angle 90
brick x 6.75 y 20 z 20
move vol 4 x 6.625
brick x 6.75 y 20 z 20
move vol 5 x -6.625
imprint all
merge all
curve 16 size 0.3
surface 12 scheme pave
mesh surface 12
volume 1 scheme auto
mesh volume 1
surface 15 scheme pave
mesh surface 15
volume 3 scheme auto
mesh volume 3
vol 4 5 scheme auto
mesh vol all
block 2 vol 1 4 5
block 1 vol 3
nodeset 1 surface 4
nodeset 2 surface 6
nodeset 3 surface 3
nodeset 4 surface 5
nodeset 11 curve 9
nodeset 12 curve 12
```

```
nodeset 13 curve 11
nodeset 14 curve 10
sideset 5 surface 22 1 16
sideset 6 surface 23 2 17
export genesis "mclan.gen"
```

File (Cylinder_Rotated.jou):

```
brick x 6.5 y 20 z 20
create Cylinder height 6.5 radius 5
rotate volume 2 about Y angle 90
subtract vol 2 from vol 1
create Cylinder height 6.5 radius 5
$rotate volume 2 about Y angle 90
rotate volume 3 about Y angle 90
brick x 6.75 y 20 z 20
move vol 4 x 6.625
brick x 6.75 y 20 z 20
move vol 5 x -6.625
imprint all
merge all
curve 16 size 0.3
surface 12 scheme pave
mesh surface 12
volume 1 scheme auto
mesh volume 1
surface 15 scheme pave
mesh surface 15
volume 3 scheme auto
mesh volume 3
vol 4 5 scheme auto
mesh vol all
block 2 vol 1 4 5
block 1 vol 3
nodeset 1 surface 4
nodeset 2 surface 6
nodeset 3 surface 3
nodeset 4 surface 5
nodeset 11 curve 9
nodeset 12 curve 12
nodeset 13 curve 11
nodeset 14 curve 10
sideset 5 surface 21
sideset 6 surface 25
export genesis "mclan.gen"
```

Appendix C. PROSCESSING

C.1 Alegra build

The build of Alegra can be accomplished by following the following steps. Once this has been accomplished one can simply re-make the build directory.

1) Build tools
a) Get the latest version of sntools (currently 1.11) from sourceforge.sandia.gov. Then extract it (tar zxvf sntools_1_11.tqz).
b) cd csource ... Which is a subdirectory of sntools
c) ./Make
d) add the sntools/engine directory to your path (e.g. set path = ($HOME/sntools/engine $path))
in .cshrc file
2) create_project -s Nevada project_name
3) cd project_name
4) checkout alegra
5) set path = ($project_dir/alegra/etc $path)
in .cshrc
6) buildTPL cvs
7) cd TPL
8) buildTPL -o serial+gnu3
9) for each library this step is only required if the debug version is required
cd $library/$version/lib
ln -s gnu2.96_opt dbg_gnu2.96
10) cd ../;mv TPL $TPL_LOCATION ..... YOU CAN SKIP THIS STEP TOO
11) build a version of alegra anywhere with
build –a < tpl_location>/TPL.xml -gi
build -a <tpl_location>/TPL.xml -o EQSSS+3D+gnu3+serial
where tpl_location is project_name/TPL
build –ig

C.2 ALEGRA Environment

In order to exicute Alegra a seris of paths and environment variables must be set as follows:
setenv ALEGRA_EXE <path_to/bin/alegra. . . x>
setenv ALEGRA_MP mp_none

C.3 Alegra Exicution

Once C1 and C2 have been accomplished, Alegra may be used by invoking the following, where run_id is the name is mclan in this case.
Alegra run_id