Spectral Methods for Modeling Microstructure Evolution in Deformation Processing of Cubic Polycrystalline Metals

A Thesis

Submitted to the Faculty

of

Drexel University

By

Hari Kishore Duvvuru

in partial fulfillment of the

requirements for the degree

of

Doctor of Philosophy

November 2007

Acknowledgments

I would like to express my sincere thanks to all those who encouraged and supported me through out my Ph.D. studies at Drexel University.

- First and foremost, I am forever indebted to my advisor and mentor, Prof. Surya Kalidindi, for his constant guidance, encouragement and financial support without which this work would not have been possible. His meticulosity and high standards of professional excellence were and will be a constant source of inspiration and challenge. His constant encouragement and motivation not only in research but also in all aspects of life are a great source of inspiration to me to have a positive outlook in general. Being his student for four years, I had the opportunity to observe first hand how passionate and committed he is in any work that he takes up, be research or administration task, I hope to carry some of his qualities in my professional career.

- I acknowledge the Office of Naval Research for funding this research work, contract No. N00014-03-1-0792 (Program Manager: Dr. Julie Christodoulou).

- My committee members Dr. Roger Doherty, Dr. Jonathan Spanier, Dr. Franco Capaldi and Dr. Joshua Houskamp for their time and feedback during my predefnese and final defense presentations.

- Dr. David Fullwood and Dr. Richrad Knight for their feedback and discussions during my PhD candidacy exam. Dr. Antonios Zavaliangos for his suggestions during my graduate seminar exam. Dr. Brent Adams for some of the discussions that I had with him during his sabbatical at Drexel.

- Faculty of department of materials Science and Engineering at Drexel University. They are always available for any discussions and especially Prof. Doherty for treating graduate students as his colleagues.

- Judith Trachtman and Dorilona Rose for their time and effort in administrative work.

- Our present and past research group memebers. Dr. Gwenaelle Proust, Dr. Joshua Houskamp, Dr. Xianping Wu (for many lunch hours that we shared and badminton games that we enjoyed) Chrisopher Hovanec, Dejan Stojakovic for sharing an office just one feet away from each other and many discussions that we had in general, Marko Knezevic for running some of the texture calculation simulations, Joshua Schafer for helping me in systems programming during the very end of my thesis work, Brendan Donhoue and Tony Fast (for constantly reminding me of my foreign accent and teaching "American slang"), Massimiliani Binci, Siddartha Pathak, Shradda Vachani, Tony Fast, Stephen Niezgoda, Naomi Barth, Giacomo Landi for keeping the graduate office environment humorous and enjoyable.

- There are really many Drexel and non-Drexel friends outside of my research group who helped me in various ways during my graduate studies. At the risk of making this acknowledgement way too lengthy, not mentioning their names explicitly here, but a very big thank you to all of them.

- A very special mention and thanks to my parents; mom-Nirmala, dad-Parasurami Reddy for their constant love and support, my brother Sathish Kumar for strongly encouraging me to go for advanced studies. Their constant moral support is invaluable for all these years of my stay in USA and in particular during four "long" years of my PhD at Drexel in Philadelphia.

Of course, in spite of all the guidance and assistance I received from many people, I alone remain responsible for the content of the following, including any errors or omissions which may unwittingly remain.

Table of Contents

List of Tables

List of Figures

Abstract

Spectral Methods for Modeling Microstructure Evolution in Deformation Processing of Cubic Polycrystalline Metals

Hari Kishore Duvvuru
Surya R. Kalidindi, Ph.D.

The mechanical properties of engineering materials are directly controlled by the underlying microstructure, which in turn is governed by the processing methods. The complete description of microstructure is extremely complex and is also not required for many microstructure-properties relationships of interest. The relevant details of the microstructure that influence strongly the elastic-plastic properties of the material include the lattice orientation distribution (texture), the grain size and shape distribution, and the arrangement and distribution of dislocation networks on the various slip systems in the constituent crystals. Of these, the crystallographic texture is perhaps the most important aspect of microstructure that has a strong influence on the elastic and the initial yield properties of most polycrystalline materials used in the manufacture of engineering components. It should also be noted that crystallographic texture is likely to have the dominant effect on the inherent anisotropy exhibited by these materials.

The objective of this thesis is to provide a mathematical framework for the development of material databases; capturing the relevant details of the microstructure, while paying attention to inherent anisotropy of properties associated with them. Here crystallographic texture is the only microstructural parameter that is considered. This work is motivated by a new design paradigm called microstructure sensitive design

(MSD) which employs statistical description of microstructure and its core feature is the efficient spectral representations of microstructure-property-processing linkages. Using the MSD framework as the basis, novel computational methodologies were developed in this work to build material spectral databases in single phase cubic polycrystalline materials. These databases were critically evaluated in three different cases: 1) To predict the effective macroscale elastic properties in perfectly disordered copper polycrystals 2) Evolution of the microstructure and the concomitant anisotropic stress-strain response during deformation processes in FCC polycrystals and (3) in developing a processing recipe to obtain a targeted texture using selected processing techniques.

CHAPTER 1: INTRODUCTION

Materials are so ubiquitous in our lives that we often overlook the revolutionary impact they have in shaping modern society. From the Bronze Age to the Silicon driven Information Age, civilization has defined itself and advanced itself by mastering new materials. Today, thanks to the availability of sophisticated software and computational resources, materials scientists are able to simulate realistically physical phenomena over a vast range of time and length scales. In this new era, extensive computational modeling will complement and reduces the number of trials in traditional methods of trial-and-error experimentation.

In the manufacture of various metallic products (e.g. automotive parts, beverage cans, steel sheet panels), metals are subjected to complex deformation processing operations such as forging, rolling, extrusion, and drawing that involve large accumulated plastic strains. The properties of a product depend significantly on its processing history and on the evolution of the underlying microstructure in the material. In order to optimize the performance of the final manufactured product, robust and reliable predictive simulation tools are necessary. Finite-element techniques (FE) have proven to be very valuable simulation tools for the prediction of material flow and stress distributions in a wide variety of metal forming processes. A very important aspect of the finite-element formulation is the model that is used to describe the material behavior. Currently, most of the FE simulations predominantly employ phenomenological constitutive theories for material behavior. Such models do not capture accurately the anisotropy inherent to the complex microstructures in polycrystalline metals, and are proving inadequate for robust design of metal forming or shaping operations. On the other hand, physics based crystal plasticity models have enjoyed remarkable success in predicting anisotropic mechanical response of several polycrystalline metals and in predicting the concurrent evolution of the underlying microstructure (mainly crystallographic texture) in finite plastic deformation. However, crystal plasticity based finite element simulation tools are extremely computationally expensive, and have therefore not yet been adopted broadly by the metal working industry.

The objective of this thesis was to develop "Microstructure Databases" to exploit material anisotropy to study complex deformation processes and provide guidance to materials design. This work was motivated by a new design paradigm called Microstructure Sensitive Design (MSD) for customized design of material microstructures for optimal performance [1-5]. MSD employs statistical description of microstructure and its core feature is the efficient spectral representation of microstructure-property-processing linkages.

This thesis is focused only on the crystallographic texture or the orientation distribution function as the main microstructural feature, and it is further restricted to single-phase, face-centered cubic metals. The organization of this thesis is as follows:

i. Section 1.1 describes crystallographic texture and its importance. Chapter 2 briefly describes spectral methods and the spectral representation of microstructure using generalized spherical harmonics as basis functions.

ii. Chapter 3 describes microstructure-property linkages in spectral framework. A novel spectral approach is presented for expressing the elastic localization tensors for polycrystalline materials with a random distribution of constituent orientations.

iii. Chapter 4 gives a brief overview of the mathematical theory of crystal plasticity. A very detailed account of the database approach for crystal plasticity is presented in chapter 5. Two new proposed methods have been critically validated with a number of case studies. Chapter 6 describes the application of microstructure databases i.e., it gives an idea of how the developed spectral databases can be used with the existing finite element tools

iv. Chapter 7 provides a case study in process design using deformation processing operations. Concluding remarks and future work are provided in Chapter 8.

1.1 Importance and Quantitative Description of Crystallographic Texture

Crystallographic texture is one of the fundamental microstructural features of all polycrystalline materials. It is described by the Orientation Distribution Function (ODF) of the crystal lattices. Its importance as a structural parameter results from the effect it has on the anisotropy of the physical properties of crystals. If the orientation distribution of the crystallites is not random, then the polycrystalline material is likely to be macroscopically anisotropic. Some of the well documented effects of anisotropy due to crystallographic texture in polycrystalline metals are

a. A polycrystalline iron sheet with the appropriate texture can be magnetically superior to a randomly oriented sheet, used in the production of the ferromagnetic sheets for transformers [6].

b. One of the widely known undesirable effects of crystallographic texture is the formation of “ears”, or nonuniform deformation in deep drawn cups [7].

c. crystallographic texture has been found to strongly influence the Langford parameter, R, (important for sheet forming operations [8, 9]).

d. crystallographic texture is also expected to strongly influence the springback phenomenon observed in sheet forming [9].

In order to define the crystallographic orientation of an individual crystallite a crystal coordinate system K_B has to be specified (e.g. given by the crystallographic axes [100] [010] [001] in case of cubic crystal). Another coordinate system K_A is related to the external shape of the sample (e.g. rolling, transverse and normal direction in rolled sheet). The crystal coordinate system K_B is related to the sample coordinate system K_A by the rotation g which specifies the orientation of the considered crystal

$$K_B = g \cdot K_A \tag{1.1}$$

The orientation g may be specified in many different ways. A very common representation is the set of Bunge-Euler angles $\{\varphi_1, \phi, \varphi_2\}$ that define a product of three rotations about specified rotation axes and the range of all possible orientations (without considerations of crystal symmetry) is established to be [6, 10].

$$0 \le \varphi_1 < 2\pi,\ 0 \le \phi \le \pi,\ 0 \le \varphi_2 < 2\pi \qquad (1.\ 2)$$

Any orientation expressed in terms of its Euler angles can be represented as a point in a three-dimensional coordinate system whose axes are given by the three Euler angles. The resulting space is referred to as Euler space [11]. The texture of the material can be characterized by the orientation distribution of the volume fraction of crystals in the orientation g.

$$\frac{dV}{V} = f(g)dg \tag{1.3}$$

ODF is normalized such that $\oint f(g)dg = 1.0$ and dg represents the invariant measure in the orientation space and is given by

$$dg = \frac{1}{8\pi^2} \sin\phi d\varphi_1 d\phi d\varphi_2 \tag{1.4}$$

Fundamental Zones of Crystal Orientation

Fundamental Zone (FZ) in the orientation space refers to the set of all physically-distinct orientations of a specific crystal system. A common choice for the FZ of cubic crystals is defined by the 'loaf' (Fig. 1) in orientation space [12]:

$$FZ(O) = \left\{ (\varphi_1, \phi, \varphi_2) \mid 0 \le \varphi_1 < 2\pi,\ \cos^{-1}\left(\frac{\cos\varphi_2}{\sqrt{1+\cos^2\varphi_2}} \right) \le \phi \le \pi/2,\ 0 \le \varphi_2 \le \pi/4 \right\}$$

(1.5)

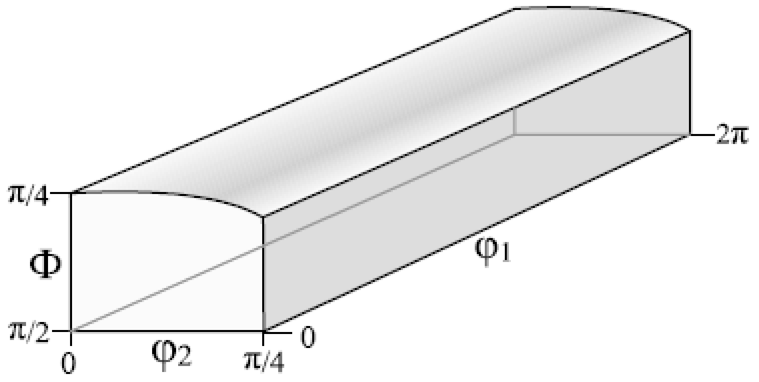


Figure 1. Fundamental zone for cubic crystal materials[12].

CHAPTER 2: SPECTRAL ANALYSIS

2.1 Introduction

Fourier methods or Spectral methods (as they're often referred to) are a class of mathematical techniques that have been employed in many fields of science and engineering to handle efficiently a very large class of computational problems. For example, in image processing, the JPEG format typically reduces the data requirements for a bitmap image by 95% [13]; in music digitization, the MP3 format compresses CD audio files by over 90%, with higher compression ratios available [14]. Both of these formats utilize a spectral representation of the original data to reduce dramatically the information storage requirements.

Spectral methods are a highly technical and well developed area of mathematics. Only the key points of spectral methods that are applicable to this work are described here; this is not intended to be an exhaustive and comprehensive review of this field. Any periodic function $y = f(x)$ with period $2l$ can be expressed as a Fourier series and can be compactly written as

$$f(x) = \sum_{n=-\infty}^{\infty} a_n e^{in\pi x/l} , \tag{2.1}$$

where the values of the Fourier coefficients a_n can be computed as:

$$a_n = \frac{1}{2l}\int_{-l}^{l} f(x) e^{-in\pi x/l} dx \;\; ; \; n = 0, \pm 1, \pm 2, ... \quad . \tag{2.2}$$

In Eq. (2.1), e^{inx} is the basis function. The choice of basis functions determines the ease or lack of it in solving the difficult problems using the spectral approaches. The three main factors that influence the choice of basis functions are:

(i) Easy to compute

(ii) Rapid convergence

(iii) Complete, which means that any solution can be represented to arbitrarily high accuracy by taking the truncation to be sufficiently large.

Decomposing a given function as in Eq. (2.1) into harmonic components helps us to manipulate the individual parts, and subsequent solution of otherwise difficult problems. Also, in many cases, use of a very few terms in the Fourier series adequately describe a complex function, resulting in an efficient approximation that reduces computing time and facilitates visualization in a low number of dimensions [12].

2.2 Spectral Representation of Orientation Distribution Function

Spectral methods have been successfully used in the field of materials science by the texture community in efficient representation of the crystallographic texture [6, 10], also called the Orientation Distribution Function (ODF). Textures in polycrystalline metals can be expressed efficiently in Fourier series as:

$$f(g) = \sum_{l=0}^{\infty} \sum_{\mu=1}^{M(l)} \sum_{\nu=1}^{N(l)} F_l^{\mu\nu} T_l^{\mu\nu}(g) \tag{2.3}$$

Where $F_l^{\mu\nu}$ represent the Fourier coefficients, and $T_l^{\mu\nu}(g)$ denote a complete set of orthonormal basis functions also referred to as generalized spherical harmonics (GSH).

For example, the ODF for a single crystal microstructure of orientation, g_k, is represented simply as $\delta(g - g_k)$, where $\delta(\;)$ denotes the Dirac-delta function [15]. Each distinct ODF can be represented by a unique set of Fourier coefficients in the infinite dimensional Fourier space. Crystal and sample symmetries can be imposed on the GSH functions and are represented by dots over $T_l^{\mu\nu}(g)$. For example if $f(g)$ possesses cubic crystal symmetry (e.g. fcc, bcc metals) and orthorhombic sample symmetry (e.g. deformation processes such as rolling, cross-rolling, simple compression, simple tension), the GSH functions would be represented as $\overset{:\,.}{T}{}_l^{mn}(g)$. Right-hand dots denote the crystal symmetry and the left-hand dots denote the sample symmetry. In case of cubic-triclinic symmetry, GSH functions are represented as $\dot{T}_l^{\mu\nu}(g)$. The functions M(*l*) and N(*l*) in Eq. (2.3) denote the number of terms needed in the enumeration of indices μ and ν; these numbers are a function of the index *l* and are determined by material and sample symmetries [6].

Some of the important properties of GSH functions and their relation with Fourier coefficients as relevant to this thesis are listed below. The reader is referred to the work of Bunge[6, 10] for more detailed analysis of these mathematical relations.

Generalized Legendre Functions

The GSH functions represented in terms of Bunge's Euler angles are given as

$$T_l^{\mu\nu}(g) = e^{i\mu\varphi_2}\, P_l^{\mu\nu}(\cos\phi)\, e^{i\nu\varphi_1} \tag{2.4}$$

In Eq. (2.4), $P_l^{\mu\nu}(\cos\phi)$ functions are generalized Legendre functions and are defined by

$$P_l^{\mu\nu}(x) = \frac{(-1)^{l-\mu}\, i^{\nu-\mu}}{2^{\ell}\,(l-\mu)!}\left[\frac{(l-\mu)!\,(l+\nu)!}{(l+\mu)!\,(l-\nu)!}\right]^{1/2} \tag{2.5}$$

$$\times (1-x)^{-\frac{\nu-\mu}{2}}\,(1+x)^{-\frac{\nu+\mu}{2}}\,\frac{d^{l-\nu}}{dx^{l-\nu}}\left[(1-x)^{l-\mu}\,(1+x)^{l+\mu}\right]$$

The $P_l^{\mu\nu}(x)$ functions are purely real if $\mu+\nu$ is even, or purely imaginary if $\mu+\nu$ is odd.

Orthonormality Condition

The symmetric GSH functions are orthonormal and when integrated over all orientations in the entire Euler space or the fundamental zone in the Euler space produce the following relation:

$$\oint \dot{\dot{T}}_l^{\mu\nu}(g)\overline{\dot{\dot{T}}_{l'}^{\mu'\nu'}}(g)dg = \frac{1}{2l+1}\delta_{ll'}\delta_{\mu\mu'}\delta_{\nu\nu'} \tag{2.6}$$

where the bar on top indicates a complex conjugate quantity, δ_{ij} represents the Kronecker delta function.

Determination of Fourier Coefficients

To determine the Fourier coefficients in Eq. (2.3), multiply both sides of Eq. (2.3) with the complex conjugate of the GSH functions and integrate over the fundamental zone in the orientation space. For example, to determine the Fourier coefficients of cubic-triclinic ODF:

$$\oint f(g)\overline{\dot{\dot{T}}_l^{\mu\nu}}(g)dg = \sum_{l=0}^{\infty}\sum_{\mu=1}^{M(l)}\sum_{\nu=1}^{N(l)} F_l^{\mu\nu}\oint \dot{\dot{T}}_l^{\mu\nu}(g)\overline{\dot{\dot{T}}_{l'}^{\mu'\nu'}}(g)dg \tag{2.7}$$

Using the orthonormality condition in Eq. (2.6), Eq. (2.7) can be simplified as:

$$F_l^{\mu\nu} = (2l+1)\oint f(g)\overline{\dot{T}_l^{\mu\nu}(g)}dg \tag{2.8}$$

If the ODF consists of only a single orientation, using the normalization condition of $f(g)$, Eq. (2.8) can be simplified as

$$F_l^{\mu\nu} = (2l+1)\overline{\dot{T}_l^{\mu\nu}(g)} \tag{2.9}$$

If the ODF consists of several different crystals with orientations g_i and volumes V_i, The Fourier coefficients are weighted average values and are given as

$$F_l^{\mu\nu} = (2l+1)\frac{\sum_i V_i \overline{\dot{T}_l^{\mu\nu}(g_i)}}{\sum_i V_i} \tag{2.10}$$

CHAPTER 3: MICROSTRUCTURE-PROPERTY LINKAGES IN A SPECTRAL FRAMEWORK

One may appreciate better the computational efficiency of the spectral representation of microstructure by linking the microstructure and their associated physical properties using the spectral methods. For a macroscopic tensorial property of interest, P, its dependence on the local crystal orientation is denoted by P(g) and expressed in Fourier series as [6, 10]

$$P(g) = \sum_{l=0}^{\infty} \sum_{\mu=1}^{M(l)} \sum_{\nu=1}^{N(l)} P_l^{\mu\nu} \dot{\ddot{T}}_l^{\mu\nu}(g) \tag{3.1}$$

The volume averaged value is then computed as [6]

$$\overline{P} = \oint P(g) f(g) dg = \sum_{l=0}^{\infty} \sum_{\mu=1}^{M(l)} \sum_{\nu=1}^{N(l)} \frac{1}{(2l+1)} P_l^{\mu\nu} F_l^{\mu\nu} \tag{3.2}$$

where $P_l^{\mu\nu}$ are referred to as the property coefficients and $F_l^{\mu\nu}$ represent the microstructure coefficients, and it is explicitly noted that this Fourier representation extends to only a finite number of terms [10, 16]. The actual number of terms required in the series is governed by the specific properties of interest. For example, it can be shown that volume averaged elastic stiffness tensor can be evaluated exactly using $l_{\max} = 4$, whereas volume averaged yield properties require consideration of a much larger number of terms [17]. Using the basic idea of microstructure-property linkage presented in Eqs. (3.1) –(3.2), elementary bounds on effective elastic stiffness parameters were established in prior work in our research group[18].

In the present work, a novel spectral approach is presented for expressing the elastic localization tensors for polycrystalline materials with a spatial distribution of constituent orientations. In particular, the formulated spectral framework is calibrated to results from finite element models. The localization tensor that relates the local elastic stress at the microscale to the macroscale (averaged) strain imposed on the composite. The proposed approach is critically validated through selected case studies in polycrystalline copper. The work that is presented in this chapter can also be found in our journal paper [19].

3.1 Elastic Localization Tensors

The primary interest here is in establishing a simple and efficient linkage for the average local stress experienced by a crystal of selected orientation *g*. Figure 2 shows a typical photomicrograph of a polycrystalline microstructure in a metallic sample. Note that crystals of a selected orientation are expected to occur in multiple locations in the microstructure, each time with a potentially different neighborhood, $N(g)$ (see Fig. 1).

In this work, $N(g)$ is assumed to be large enough to take into account all details of the neighborhood that influence the local stress in the crystal of interest. Let each occurrence of the crystal of the selected orientation be associated with a neighborhood $N(g)^{(k)}$ and experience an average local stress $\sigma_{ij}(g)^{(k)}$. We seek here the average stress experienced by all of the crystals of a given orientation, and denote it as $\sigma_{ij}(g)$. The main hypothesis here is that $\sigma_{ij}(g)$ can be linked to the imposed macroscale strain tensor as

$$\sigma_{ij}(g) = a_{ijkl}(g, \langle N(g) \rangle) \langle \varepsilon_{kl} \rangle, \tag{3.3}$$

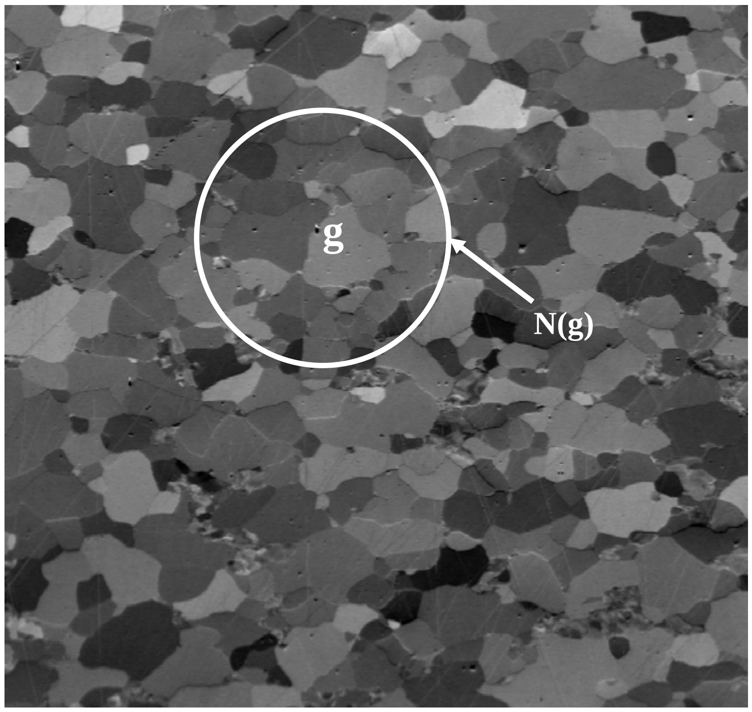


Figure 2. A photomicrograph depicting a typical polycrystalline microstructure in a metallic sample. The meaning of the neighborhood N(*g*) used in this study is conveyed in this figure.

where $\langle N(g) \rangle$ denotes the ensemble averaged neighborhood of the orientation of interest, and a_{ijkl} is the fourth-rank stress localization tensor. Eq. (3.3) was largely motivated by the well established generalized composite theories [20-22].

Note that the statistical details of the $\langle N(g) \rangle$ can be described quantitatively using the formalism of *n*-point statistics [23]. For example, the 1-point statistics of $\langle N(g) \rangle$ is essentially the texture in the sample (given by the Orientation Distribution Function as described in Chapter 1). The ODF is denoted by $f(g')$ and reflects the normalized probability density associated with the occurrence of crystallographic orientation g' in $\langle N(g) \rangle$. Likewise, the 2-point statistics of $\langle N(g) \rangle$ can be represented by $f_2(g', g'' | \mathbf{r})$ that reflects the normalized probability density associated with the ordered occurrences of orientations g' and g'' in $\langle N(g) \rangle$ separated by a specified vector **r**.

Using this formalism, Eq. (3.3) can be recast as

$$\sigma_{ij}(g) = a_{ijkl}\left(g, f(g'), f_2(g', g'' | \mathbf{r}), f_3(g', g'', g''' | \mathbf{r}, \mathbf{r}'), \ldots\right)\langle \varepsilon_{kl} \rangle . \tag{3. 4}$$

In Eq. (3. 4), the stress localization tensor is written in its most general form. The functional dependence of a_{ijkl} on the details of $\langle N(g) \rangle$ can take on extremely complex forms. Use of appropriate Fourier representations for the *n*-point statistics allows us to recast Eq. (3.4) as

$$\sigma_{ij}(g) = a_{ijkl}\left(g, F_L^{\mu\nu}, \ldots\right)\langle \varepsilon_{kl} \rangle . \tag{3.5}$$

The dependence of $a_{ijkl}\left(g, F_L^{\mu\nu}, \ldots\right)$ on the orientation g can also be captured efficiently in a Fourier series using GSH functions as

$$a_{ijkl}(g) = \sum_L \sum_\mu \sum_\nu {}_{ijkl}A_L^{\mu\nu}\left(F_{L'}^{\mu'\nu'}, \ldots\right)\dot{T}_L^{\mu\nu}(g) \tag{3.6}$$

The problem at hand then reduces to establishing the functional form of ${}_{ijkl}A_L^{\mu\nu}\left(F_{L'}^{\mu'\nu'}, \ldots\right)$. Note also that the effective macroscale elastic stiffness tensor can be computed as

$$C^*_{ijkl} = \int a_{ijkl}\left(g, F_L^{\mu\nu}, \ldots\right) f(g)\, dg = \sum_L \sum_\mu \sum_\nu \frac{{}_{ijkl}A_L^{\mu\nu}\left(F_{L'}^{\mu'\nu'}, \ldots\right)F_L^{\mu\nu}}{(2L+1)} . \tag{3.7}$$

An important feature of the spectral representations described above is that it can be proven that the coefficients ${}_{ijkl}A_L^{\mu\nu}$ are zero valued for $L > 4$. This feature results in tremendous efficiency in quantifying ${}_{ijkl}A_L^{\mu\nu}\left(F_{L'}^{\mu'\nu'},...\right)$ in the minimum number of possible terms. Consequently, we will not need all of the microstructure coefficients $F_L^{\mu\nu}$; as only a finite number of these will impact the computation of the average local stress or the effective stiffness tensor using Eqs. (3.5)-(3.7).

3.2 Case Study: Perfectly Disordered Copper Polycrystals

A case study to illustrate the main strategy for assembling the database of ${}_{ijkl}A_L^{\mu\nu}$ coefficients is presented in this section. As a first demonstration of the proposed framework, we restrict our attention here to perfectly disordered random textures in copper polycrystals. For this special class of microstructures, the Fourier coefficients describing the *n*-point statistics are fully prescribed and equal to zero in the generalized spherical harmonic space. Let

$$ {}_{ijkl}^{\;\;0}A_L^{\mu\nu} = {}_{ijkl}A_L^{\mu\nu}\left(0,0,...\right) \tag{3.8}$$

denote the needed coefficients for describing the stress localization tensor for the perfectly disordered random textures in copper polycrystals.

The basic strategy for establishing the ${}_{ijkl}^{\;\;0}A_L^{\mu\nu}$ is to consider special loading conditions that conveniently decouple the coefficients. For example, if we impose an uniaxial strain at the macroscale on the perfectly disordered polycrystal, say $\langle \varepsilon_{11} \rangle = \varepsilon_o$ with all other $\langle \varepsilon_{ij} \rangle = 0$, then Eqs. (3.5) and (3.7) reduce to

$$ {}^{o}\sigma_{ij}(g) = {}^{o}a_{ij11}(g)\varepsilon_o , \quad {}^{o}a_{ij11}(g) = \sum_{L}\sum_{\mu}\sum_{\nu} {}_{ij11}^{\;o}A_L^{\mu\nu}\,\dot{T}_L^{\mu\nu}(g), \tag{3.8} $$

where the left superscript *o* reminds us that we are dealing with the perfectly disordered polycrystal with a random texture. Using the standard methods of Fourier analyses described in chapter 2 allow us to express the coefficients of interest as

$$ {}_{ij11}^{\;o}A_L^{\mu\nu} = \frac{(2L+1)}{\varepsilon_o}\int_{FZ} {}^{o}\sigma_{ij}(g)\overline{\dot{T}_L^{\mu\nu}}(g)dg\,. \tag{3.9} $$

FZ denotes the fundamental zone of distinct orientations for the selected class of textures. Using Eq. (3.9), ${}_{ij11}^{\;o}A_L^{\mu\nu}$ can be computed numerically (e.g. the Simpson method [24]) if the local stresses in grains of different orientations distributed throughout the FZ are known. In the approach proposed here, these local stresses are obtained from a micromechanical finite element model.

Figure 3 shows a finite element model of a copper polycrystal developed in this study for extracting the information mentioned above. This model comprised 50653 8-noded three-dimensional solid elements (C3D4 elements; ABAQUS [25]), where each element is assumed to represent one single crystal. The local single crystal elastic stiffness in each element is defined as

$$ \mathrm{C}_{\mathrm{abcd}}(g) = \mathrm{C}_{12}\delta_{\mathrm{ab}}\delta_{\mathrm{cd}} + \mathrm{C}_{44}(\delta_{\mathrm{ac}}\delta_{\mathrm{bd}} + \delta_{\mathrm{ad}}\delta_{\mathrm{bc}}) + (\mathrm{C}_{11} - \mathrm{C}_{12} - 2\mathrm{C}_{44})\sum_{\mathrm{r}=1}^{3} \mathrm{g}_{\mathrm{ar}}\mathrm{g}_{\mathrm{br}}\mathrm{g}_{\mathrm{cr}}\mathrm{g}_{\mathrm{dr}} \tag{3.10} $$

where g_{ij} represent the components of the transformation matrix from the local crystal reference frame to the global (sample) reference frame. For copper, C_{11} = 168.4 GPa, C_{12} = 121.4 GPa, and C_{44} = 75.4 GPa [26].

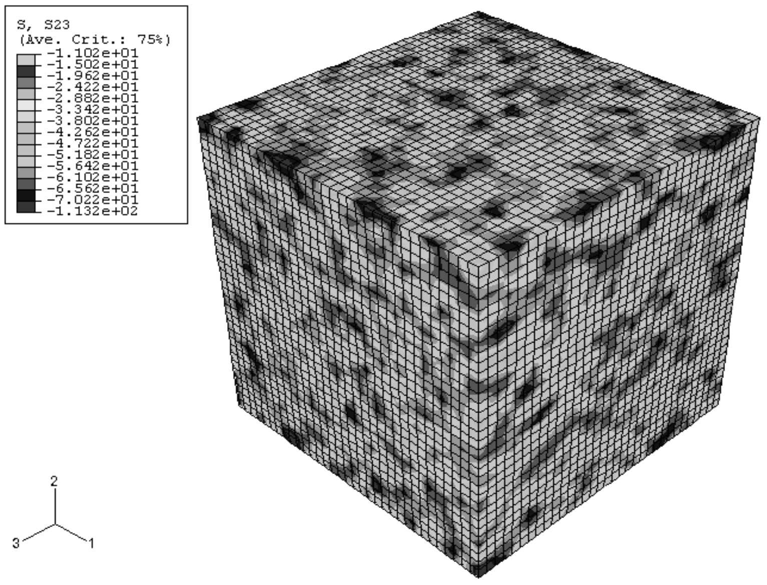


Figure 3. A micromechanical finite element model of a copper polycrystal subjected to a macroscale shear strain in 2-3 plane. Each element in the finite element mesh is assumed to represent one single crystal. The contours depicted are for the local shear stress, σ_{23}.

A set of 35301 crystal orientations were selected in the FZ (see Eq. (1.5)) such a way that they facilitated a convenient numerical evaluation of the integral in Eq. (3.9) using a Simpson method. Using this basic set of 35301 orientations, we created a set of 50653 orientations by repeating some of the orientations in the original set in such a way that the final texture for the set of 50653 orientations was as close to a random texture as possible. The set of 50653 orientations were then assigned to the 50653 elements of the finite element model in two steps: (i) First, the basic 35301 orientations contained in the set of the 50653 orientations were assigned randomly to the interior elements of the finite element model (an element is considered an interior element if it is separated from the boundary by at least two other elements). (ii) Second, the leftover 15352 orientations were assigned randomly to the remaining elements of the mesh. This process accomplished two goals. First, the set of 35301 orientations needed in numerical integration of Eq. (3.9) did not constitute a random texture, but the set of 50653 orientations did constitute a random texture. Second, it allowed us to embed the set of 35301 orientations of interest in the interior of the finite element model. The model was then subjected to boundary conditions that produced the desired uniaxial strain at the macroscale. The local stresses in the interior elements computed by the finite element model were mapped back to the assigned crystal orientations in these elements and stored. The procedure described above was repeated 10 times, each time with a different but random assignment of the crystal orientations to the elements of the finite element model. The stress results for each crystal orientation from the different finite element runs were averaged to take care of any local texturing effects and then used in Eq. (3.9) to obtain the ${}_{ij11}^{\ o}A_L^{\mu\nu}$. To establish the complete set of coefficients ${}_{ijkl}^{\ 0}A_L^{\mu\nu}$, the procedure described above was repeated for a total of six different strain states, where each state was defined with a single non-zero strain component (three normal strains and three shear strains).

The values of selected ${}_{ijkl}^{\ 0}A_L^{\mu\nu}$ coefficients computed for copper polycrystals are shown in Table 1 (note that the ${}_{ijkl}^{\ 0}A_L^{\mu\nu}$ coefficients are complex numbers). To validate the ${}_{ijkl}^{\ 0}A_L^{\mu\nu}$ coefficients, we created another finite element model comprising of 1000 elements. A set of 1000 crystal orientations corresponding to a random texture were produced and assigned randomly to the elements of the validation finite element model. The predictions for the overall stiffness from the validation finite element model are compared against the predictions from the spectral linkage developed here are presented in Table 2. The components of the volume averaged stiffness tensor and the inverse of the volume averaged compliance tensor (these make appearances in the elementary bounding theories) are also presented in this table. It is seen that the predictions from the finite element model are bracketed by the elementary bounds and that the predictions from the spectral linkages developed here are in excellent agreement with the finite element predictions.

Table 1: Values of selected ${}^{0}_{ijkl}A_L^{\mu\nu}$ coefficients for copper polycrystals computed using the spectral framework outlined in this paper. The units of these coefficients are GPa.

	${}^{0}_{ijkl}A_L^{\mu\nu}$					
$(L\mu\nu)$	${}^{0}_{1111}A$	${}^{0}_{2222}A$	${}^{0}_{3333}A$	${}^{0}_{2323}A$	${}^{0}_{1313}A$	${}^{0}_{1212}A$
000	205.16	205.16	204.83	51.44	50.95	50.81
411	-164.19	-163.58	-442.65	-4.81	20.00	19.92
412	0.03 - 0.03 i	-0.15 - 0.02 i	0.02 i	-0.04 + 0.01 i	0.03 i	0.02 - 0.02 i
413	-0.03 - 0.02 i	0.01 - 0.02 i	0.02 i	0.04 + 0.01 i	0.03 i	-0.02 - 0.02 i
414	177.20 + 0.07 i	-177.05 - 0.02 i	0.05 - 0.53 i	0.03 - 0.01 i	-157.34	15.59 - 0.03 i
415	177.20 - 0.07 i	-177.05 + 0.02 i	0.05 + 0.53 i	0.03 + 0.01 i	-157.34	15.59 + 0.03 i
416	0.03 + 0.14 i	-0.06 - 0.06 i	-0.06 - 0.07 i	0.02 i	-0.03 - 0.01 i	-0.03 - 0.02 i
417	-0.03 + 0.14 i	0.06 - 0.06 i	-0.06 + 0.07 i	0.025 i	0.03 - 0.01 i	0.03 - 0.02 i
418	-232.16 - 0.06 i	-232.34 + 0.04 i	0.23 + 0.02 i	208.50 - 0.02 i	0.13 - 0.06 i	0.08 - 0.02 i
419	-232.16 + 0.06 i	-232.34 - 0.04 i	0.23 - 0.02 i	208.50 + 0.02 i	0.13 + 0.06 i	0.08 + 0.02 i

Table 2: The predictions of the components of overall stiffness from the validation finite element model and the spectral linkages developed in this paper (units are in gpa). The components of the volume averaged stiffness tensor ($\overline{C}_{ijkl}$) and the inverse of the volume averaged compliance tensor ($\overline{S}_{ijkl}^{-1}$) are also presented in this table.

	1111	2222	3333	2323	1313	1212	1122	1133
$\overline{C}_{ijkl}$	210.0	210.0	210.0	54.7	54.6	54.6	100.7	100.6
$C^*_{ijkl}\big\vert$ Spectral method	204.9	204.9	205.3	51	51.7	51.8	102.7	103
$C^*_{ijkl}\big\vert$ Validation FEM model	205.2	205.2	204.9	51.4	50.9	50.8	102.8	102.9
$\overline{S}_{ijkl}^{-1}$	192.3	192.3	192.4	40.1	40	40	112.2	112.1

As a final check we also compared the predictions of the local stresses from both the finite element models and the spectral linkages developed here (Eq. (3.9)). From the first validation model involving a copper polycrystal with an overall random texture and subjected to an uniaxial strain along the **e**$_1$-axis, sorted all of the 1000 crystal orientations in the order of increasing σ_{11} values. In Table 3, the five crystal orientations that produced the highest σ_{11} values, the five crystal orientations that produced the intermediate σ_{11} values, and the five crystal orientations that produced the lowest σ_{11} values, according to the predictions from the spectral linkages developed in this study are presented. Also shown in the table for comparison are the local stresses in the same selected crystals predicted from the validation finite element model. The finite element predictions are shown as an average with a standard deviation after running the finite element model 10 times, where each run represents a different but random assignment of the same set of crystal orientations to the elements of the finite element model. It is clearly seen from Table 3 that the spectral linkages developed here track extremely well the orientation dependence of local stresses in individual crystals.

Table 3. Predictions of the local crystal stresses (σ_{11}) experienced by individual crystals from the finite element models and the spectral method. The overall texture in the validation finite element model represents a random texture.

Phi1	Phi	Phi2	Spectral	FEM
Crystals with high stresses				
40.5	61.4	21.5	228.4	225.1 ± 7
220.5	70.9	30.5	228.1	232.8 ± 7.7
220.5	61.4	21.5	228.0	221.6 ± 8.4
40.5	70.9	30.5	228.0	232.1 ± 14.3
49.5	50.4	4.3	227.9	224.6 ± 7.9
Crystals with medium stresses				
193.5	53.9	30.5	213.9	213.3 ± 9.5
211.5	77.9	12.8	213.9	211.1 ± 9.6
76.5	50.9	12.8	213.9	216.9 ± 9.1
139.5	69.3	4.3	213.9	213.9 ± 10.9
301.5	69.3	4.3	213.9	212.1 ± 14.3
Crystals with low stresses				
274.5	85.9	4.3	174.0	171.5 ± 8.7
94.5	85.9	4.3	173.9	177.3 ± 10
265.5	85.9	4.3	173.9	172.4 ± 7.7
274.5	86.0	12.8	173.9	179.6 ± 5
94.5	86.0	12.8	173.9	176 ± 12.4

CHAPTER 4: CRYSTAL PLASTICITY

Crystal Plasticity is the study of plastic deformation in single crystal and polycrystalline materials while taking into account explicitly the details of physics and geometry of deformation at the crystal (also called grain) level.

In metals, at low homologous temperatures, crystallographic slip and deformation twinning are the primary observed modes of plastic deformation. Both these modes of plastic deformation take place on characteristic planes, i.e. slip planes or twinning planes, in characteristic directions, i.e. slip directions or twinning directions. The orientations of the characteristic planes and directions are a function of the geometry of the crystal. The present work addresses plastic deformation of FCC crystals by crystallographic slip alone. As a crystal deforms by slip, it undergoes lattice rotation. These lattice rotations are the cause of development of preferred orientations in polycrystalline metals. In FCC crystals, such as Copper, Aluminum and Nickel, the slip planes are close packed octahedral planes (Fig. 4). There are three slip directions in each slip plane, along the diagonals of the cube planes. As there are four different orientations of slip planes, there are therefore a total of 12 possible slip systems which can take part during the plastic deformation. The slip systems are defined using standard Miller index notation by (111) planes and {110} directions and the 12 possible slip systems in FCC crystals are listed in table 4.

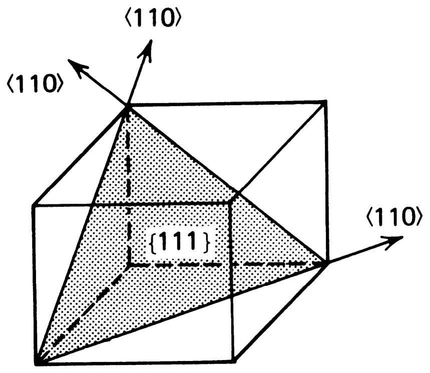


Figure 4. Schematic of typical slip systems of face-centered cubic crystals [26].

Table 4. Slip systems in FCC crystals

	Slip Plane	Slip Direction
1.	{1 1 1}	[0 1 $\bar{1}$]
2.	{1 1 1}	[1 0 $\bar{1}$]
3.	{1 1 1}	[1 $\bar{1}$ 0]
4.	{1 1 $\bar{1}$}	[0 1 1]
5.	{1 1 $\bar{1}$}	[1 0 1]
6.	{1 1 $\bar{1}$}	[1 $\bar{1}$ 0]

7.	$\{1\,\bar{1}\,\bar{1}\}$	$[0\,1\,\bar{1}]$
8.	$\{1\,\bar{1}\,\bar{1}\}$	$[1\,0\,1]$
9.	$\{1\,\bar{1}\,\bar{1}\}$	$[1\,1\,0]$
10.	$\{1\,\bar{1}\,1\}$	$[0\,1\,1]$
11.	$\{1\,\bar{1}\,1\}$	$[1\,0\,\bar{1}]$
12.	$\{1\,\bar{1}\,1\}$	$[1\,1\,0]$

In general, all polycrystal plasticity models are made up of two parts: a set of crystal equations describing properties and orientations, and a set of equations that link individual crystals together into a polycrystal. The next two sections describe the crystal plasticity kinematic framework and some of the polycrystal averaging schemes.

4.1 Single Crystal Kinematics

The deformation behavior of grains is determined by a crystal plasticity model that accounts for plastic deformation by crystallographic slip and for the rotation of the crystal lattice during deformation. The crystal plasticity modeling framework [27] used in this study is briefly summarized here using a notation that is now standard in modern continuum mechanics textbooks. For finite deformations, the total deformation gradient tensor (**F**) can be decomposed into elastic and plastic components as [28]

$$\mathbf{F} = \mathbf{F}^{*}\,\mathbf{F}^{\mathrm{p}} \tag{4.1}$$

where $\mathbf{F}^*$ contains deformation gradients due to both elastic stretching as well as the lattice rotation, while $\mathbf{F}^{\mathrm{p}}$ denotes the deformation gradient due to plastic deformation alone. The constitutive equation for stress in the crystal can be expressed as [27]:

$$\mathbf{T}^* = \mathbf{C}\left[\mathbf{E}^*\right], \quad \mathbf{T}^* = \mathbf{F}^{*-1}\left\{\left(\det\ \mathbf{F}^*\right)\mathbf{T}\right\}\mathbf{F}^{*-\mathrm{T}}, \quad \mathbf{E}^* = \frac{1}{2}\left\{\mathbf{F}^{*\mathrm{T}}\mathbf{F}^* - \mathbf{1}\right\} \tag{4.2}$$

where $\mathbf{C}$ is the fourth-order elasticity tensor, $\mathbf{T}^*$ and $\mathbf{E}^*$ are a pair of work conjugate stress and strain measures, and $\mathbf{T}$ is the Cauchy stress in the crystal. The evolution of $\mathbf{F}^{\mathrm{p}}$ can be expressed in a rate form as

$$\dot{\mathbf{F}}^{\mathrm{p}} = \mathbf{L}^{\mathrm{p}}\mathbf{F}^{\mathrm{p}}, \quad \mathbf{L}^{\mathrm{p}} = \sum_{\alpha}\dot{\gamma}^{\alpha}\mathbf{S}_{\mathrm{o}}^{\alpha}, \qquad \mathbf{S}_{\mathrm{o}}^{\alpha} = \mathbf{m}_{\mathrm{o}}^{\alpha} \otimes \mathbf{n}_{\mathrm{o}}^{\alpha}, \tag{4.3}$$

where $\dot{\gamma}^{\alpha}$ is the shearing rate on the slip system α, and $\mathbf{m}_{\mathrm{o}}^{\alpha}$ and $\mathbf{n}_{\mathrm{o}}^{\alpha}$ denote the slip direction and the slip plane normal of the slip system, respectively, in the initial unloaded configuration. The shearing rate on the slip system is dependent on the resolved shear (τ^{α}) on the slip system and the slip resistance (s^{α}) of that slip system, and can be expressed in a power-law relationship [29]:

$$\dot{\gamma}^{\alpha} = \dot{\gamma}_{\mathrm{o}}\left|\frac{\tau^{\alpha}}{s^{\alpha}}\right|^{1/m}\mathrm{sgn}(\tau^{\alpha}), \qquad \tau^{\alpha} \approx \mathbf{T}^* \bullet \mathbf{S}_{\mathrm{o}}^{\alpha} \tag{4.4}$$

In Eq. (4.4), $\dot{\gamma}_{\mathrm{o}}$ denotes a reference value of the slip rate (taken here as 0.001 sec^{-1}) and *m* represents the strain rate sensitivity parameter (taken here as 0.01 to capture behavior of most metals at low homologous temperatures). In the present study, for simplicity, we have adopted a simple saturation type hardening law to describe the evolution of the slip resistances:

$$\dot{s}^{\alpha} = h^{o}\left(1-\frac{s^{\alpha}}{s^{s}}\right)^{a}\sum_{\beta}\left|\dot{\gamma}^{\beta}\right|. \qquad (4.5)$$

The material parameters in Eq. (4.5) for annealed OFHC copper were estimated by curve fitting the Taylor predictions to experimental measurements in a previous study [27]. The values obtained were $h^{o} = 180$ MPa, $s^{s} = 148$ MPa, $a = 2.25$. Note that the hardening law used here does not reflect any latent hardening (i.e. it assumes equal hardening of all slip systems). The lattice spin $\mathbf{W}^{*}$ in the crystal is given by

$$\mathbf{W}^{*} = \mathbf{W}^{app} - \mathbf{W}^{p}, \qquad \mathbf{W}^{p} = \frac{1}{2}\left(\mathbf{L}^{p} - \mathbf{L}^{pT}\right), \qquad (4.6)$$

The numerical procedures for the integration of this constitutive model have been described in detail in prior literature [27].

4.2 Polycrystal Homogenization Theories

The polycrystal constitutive theory of plasticity treats the plastic properties of a polycrystalline aggregate as averages over all of the single crystals. Several different approaches have been taken in the literature for modeling the behavior of polycrystal materials. Most of the polycrystalline simulations have used highly simplistic approximations, e.g. that each grain deforms homogeneously, and yet arrived, at least for moderate strains, at useful results. There are two extreme assumptions, one proposed by Sachs [30] and the other proposed by Taylor [31]. Sachs proposed that the single crystals experience the same state of stress as the aggregate. This condition satisfies equilibrium across the grain boundaries, but does not satisfy compatibility between the grains and

Sachs model leads to inaccurate texture predictions compared to that of the experimental results. Taylor proposed a simple isostrain model (strains are the same in all grains and are equal to the macroscopic strain) for predicting the macroscopic yield strength of polycrystals, and this model with relatively minor improvements continues to be the most widely used theory even today. Based on geometric considerations, a modified version of the Taylor model that relaxes some components of the imposed deformation was developed by Kocks and Canova [32]. The theory of Taylor was expanded upon by Bishop and Hill to include multi-axial stress states and yielded the Taylor-Bishop-Hill model [33]. Later, the Taylor model was extended to viscoplastic deformations by Hutchinson [34] and to finite elastic-viscoplastic deformations by Asaro and Needleman [29]. All of them are generally referred to as the Taylor-type models. In these models, each individual grain in the representative volume element (RVE) of the polycrystal is treated completely independent of all the other grains in the polycrystal.

The other type of models are micro-mechanical finite element (FE) models which take the grain interactions into account in modeling the deformations of an aggregate of polycrystals [27, 35-38]. In these micromechanical models, the weak form of the principle of virtual work (including equilibrium and the boundary conditions) is satisfied in a given domain, discretized into a finite element mesh, by evaluating the constitutive response at specific locations in each element called integration points. A polycrystalline aggregate is discretized into a finite element mesh, such that each grain is modeled by one or more finite elements to allow for non-uniform deformations between the grains and within the grains. Micro-mechanical finite element models are, in principle, the best models for the simulation of plastic deformation of polycrystalline materials and for the prediction of deformation textures.

Both the Taylor-type models and micro-mechanical FE models have been extensively validated by direct comparisons with experiments by several research groups [39-46]. It is now well accepted in literature that the Taylor-type crystal plasticity models provide good predictions for the anisotropic stress-strain curves exhibited by cubic metals along with the evolution of the underlying crystallographic texture in a broad range of deformation paths, although the texture predictions are usually much sharper than the experimentally measured ones. The predictions from micro-mechanical FE models tend to predict more diffuse textures that are in better agreement with experiments, but continue to lack the desired accuracy at the scale of individual single crystals [44, 47]. Although the crystal plasticity models have shown tremendous potential in capturing well the overall anisotropic response of polycrystalline materials subjected to a broad range of deformation paths while also capturing the important details of the evolution of the averaged crystallographic texture in the sample, their usage has been limited to only a small number of studies in simulations of deformation processing operations in the metal shaping industry.

The main factor hindering the much wider use of these models is the high computational cost associated with them; the requirement for high computational resources persists in spite of the successful development and implementation of very efficient numerical schemes for these calculations. For example, in a finite element simulation of the deep drawing of a car body panel, several tens of thousands of elements need to be used, each representing an entire polycrystal [48]. It is practically impossible to use crystal plasticity finite element method for each of them. Consequently, there are several ongoing efforts in current literature to develop novel techniques that would possess the predictive capabilities of the Taylor-type model, but would require significantly less computational resource [38, 49-51]. In many of these approaches, the underlying physics governing the evolution of the crystallographic texture in deformation processes continues to be provided by the Taylor-type model (actually the proposed methods are often calibrated with Taylor-type model to ensure good agreements).

CHAPTER 5: DATABASE APPROACH FOR CRYSTAL PLASTICITY

As remarked earlier in chapter 4, the current crystal plasticity models demand an extremely high computational effort. Furthermore, one finds in crystal plasticity simulations of deformation processing operations that same or similar computations are repeated several times given that every material point under consideration has many individual crystals, and there are several points of interest in the deformation processing operation. This raises the obvious question of whether it would be possible to capture the solutions from the crystal plasticity models in an efficient representation, such that one might do the necessary computations once and store the results in an efficient database that can be used to perform all subsequent computations. Towards this objective, two different mathematical frameworks namely: (1) Bunge-Esling approach and (2) Spectral Crystal Plasticity has been formulated. Detailed descriptions of these two approaches can be found in our published papers [52, 53]. However, to make this thesis report as self contained as possible, key features of these two approaches from those two papers [52, 53] are reproduced here.

5.1 Bunge-Esling Approach

About two decades ago, Bunge and Esling [54] put forth a novel concept for capturing and simulating crystallographic texture evolution during large plastic strains on metals using an efficient spectral representation of the orientation distribution function. Although this methodology indicated promise, it was never evaluated critically by these authors, possibly because of the high demands it placed on computational resources. In this thesis, the Bunge and Esling concept is revisited and evaluated critically for the first time for a range of deformation processes and starting textures.

Utilizing the spectral representations of the ODF described in chapter 2 together with a framework proposed by Clement and Coulomb [55] for representation of crystal lattice rotations as a vector flow field in an eulerian angle space, Bunge and Esling [54] derived the following simple relation to represent the evolution of texture in a given deformation process (with a prescribed macroscopically imposed velocity gradient tensor):

$$\frac{dF_l^{\mu\nu}}{d\eta} = \sum_{\lambda=0}^{\infty} \sum_{\sigma=1}^{M(\lambda)} \sum_{\rho=1}^{N(\lambda)} \tilde{A}_{l\lambda}^{\mu\nu\sigma\rho} F_{\lambda}^{\sigma\rho} \tag{5.1}$$

In Eq. (5.1), η is an appropriate metric of the deformation process (for example, for rolling η could represent the compressive strain), and the coefficients $\tilde{A}_{l\lambda}^{\mu\nu\sigma\rho}$ are constant for a given deformation process independent of the crystal lattice orientation. Following the original derivation of Eq. (5.1) by Bunge and Esling [54], one can recognize that the coefficients $\tilde{A}_{l\lambda}^{\mu\nu\sigma\rho}$ are essentially Fourier coefficients in the spectral representation of the texture flow field in the orientation space for a given deformation process.

As with any spectral representation, one can indeed write an analytical expression for these coefficients. However, as remarked by Bunge and Esling [54], the derivation of an analytical expression for $\tilde{A}_{l\lambda}^{\mu\nu\sigma\rho}$ requires the use of tedious calculus involving field theory and Clebsch-Gordon coefficients. To circumvent this challenge, Bunge and Esling [54] suggest that the coefficients $\tilde{A}_{l\lambda}^{\mu\nu\sigma\rho}$ be established numerically for a specified deformation process by calibrating Eq. (5.1) with predictions from a Taylor-type model for a large number of single crystal orientations. The calibration procedure suggested by these authors is essentially a linear regression analysis to find the coefficients $\tilde{A}_{l\lambda}^{\mu\nu\sigma\rho}$ by minimizing the error in Eq. (5.1) for a very large number of single crystal orientations, for a given deformation process. Although they describe this numerical approach for the computation of $\widetilde{A}_{l\lambda}^{\mu\nu\sigma\rho}$ in detail, they actually do not report performing such a calculation or any results of such a calculation. They specifically cite that for expansions of the Fourier series to include terms up to *l = 22* for cubic orthorhombic ODF, one would need to handle 185 $F_{l}^{\mu\nu}$ terms and 34,225 (= 185^2) independent $\widetilde{A}_{l\lambda}^{\mu\nu\sigma\rho}$ terms. In this work, we have included the terms in the Fourier expansion up to *l = 17*, resulting in 83 independent $F_{l}^{\mu\nu}$ terms and 6972 (= 83^2) independent $\widetilde{A}_{l\lambda}^{\mu\nu\sigma\rho}$ terms.

5.1.1 Numerical Procedure

Recognizing that F_{0}^{11} is always equal to 1.0 [6], it was found convenient to recast Eq. (5.1) as

$$F_{l}^{\mu\nu}(\Delta\eta) = \sum_{\lambda=1}^{\infty}\sum_{\sigma=1}^{M(\lambda)}\sum_{\rho=1}^{N(\lambda)} A_{l\lambda}^{\mu\nu\sigma\rho} F_{\lambda}^{\sigma\rho}(0) + B_{l}^{\mu\nu} \tag{5.2}$$

In Eq. (5.2), $\Delta\eta$ is a suitable increment of the specific deformation process being modeled, $F_{\lambda}^{\sigma\rho}(0)$ denote the Fourier coefficients of the initial ODF, $F_{l}^{\mu\nu}(\Delta\eta)$ denote the Fourier coefficients of the ODF after an incremental deformation of $\Delta\eta$, and $A_{l\lambda}^{\mu\nu\sigma\rho}$ and $B_{l}^{\mu\nu}$ are constants for a given deformation process that are independent of the texture. Furthermore, note that in Eq. (5.2), $\lambda = 0, \sigma = 1, \rho = 1$ is excluded (because F_{0}^{11} is always equal to 1.0). Finally, it is implicit in Eq. (5.2) that it can be used in a recursive manner:

$$F_l^{\mu\nu}(N\Delta\eta) = \sum_{\lambda=1}^{\infty}\sum_{\sigma=1}^{M(\lambda)}\sum_{\rho=1}^{N(\lambda)} A_{l\lambda}^{\mu\nu\sigma\rho}\, F_\lambda^{\sigma\rho}\left((N-1)\Delta\eta\right) + B_l^{\mu\nu}\ , \qquad (5.3)$$

where N is an integer and $A_{l\lambda}^{\mu\nu\sigma\rho}$ and $B_l^{\mu\nu}$ are constants independent of η.

To establish the constants $A_{l\lambda}^{\mu\nu\sigma\rho}$ and $B_l^{\mu\nu}$, we selected a set of 1000 single crystal orientations that are distributed uniformly in the fundamental zone of the cubic-orthorhombic Euler space, where each of these orientations is represented by $g^{(k)}$. For each of these orientations, the lattice orientation after an incremental strain in a selected deformation process is computed using the Taylor-type model described in chapter 4. The corresponding final orientation is denoted by $g^{(k)'}$. The deformation itself is defined through the velocity gradient tensor **L**. Two deformation processes: (i) plane strain compression (PSC), and (ii) simple compression (SC) are investigated. The corresponding velocity gradient tensors are given as:

$$\mathbf{L}_{PSC} = \begin{bmatrix} -\dot{\varepsilon} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \dot{\varepsilon} \end{bmatrix}, \qquad \mathbf{L}_{SC} = \begin{bmatrix} -\dfrac{\dot{\varepsilon}}{2} & 0 & 0 \\ 0 & -\dfrac{\dot{\varepsilon}}{2} & 0 \\ 0 & 0 & \dot{\varepsilon} \end{bmatrix} \qquad (5.4)$$

For each of the selected 1000 grains in this analyses, the Fourier coefficients corresponding to the initial orientation, $\left(F_l^{\mu\nu}(0)\right)^{(k)}$, and those corresponding to the orientation predicted by the Taylor-type model after an incremental strain in the selected deformation process, $\left(F_l^{\mu\nu}(\Delta\eta)\right)^{(k)'}$, are computed. A χ^2 technique has been used to extract the values of the constants $A_{l\lambda}^{\mu\nu\sigma\rho}$ and $B_l^{\mu\nu}$ that would provide the best possible fit to Eq. (5.3) for all 1000 single crystal orientations selected. For this purpose, an error function is defined as:

$$\chi^2 = \sum_{k=1}^{1000}\sum_{l=1}^{\infty}\sum_{\mu=1}^{M(l)}\sum_{\nu=1}^{N(l)}\left(\left(F_l^{\mu\nu}(\Delta\eta)\right)^{(k)'} - \sum_{\lambda=1}^{\infty}\sum_{\sigma=1}^{M(\lambda)}\sum_{\rho=1}^{N(\lambda)} A_{l\lambda}^{\mu\nu\sigma\rho}\left(F_\lambda^{\sigma\rho}(0)\right)^{(k)} - B_l^{\mu\nu} \right)^2 \qquad (5.5)$$

In the χ^2 technique, the error function is minimized by setting its derivatives with each of the desired parameters (constants $A_{l\lambda}^{\mu\nu\sigma\rho}$ and $B_l^{\mu\nu}$) to zero. Because Eq. (5.5) is quadratic in these parameters, the derivatives are linear, and therefore this technique results in a set of simultaneous linear equations that can then be solved for the desired parameters $A_{l\lambda}^{\mu\nu\sigma\rho}$ and $B_l^{\mu\nu}$.

5.1.2 Computation of $A_{l\lambda}^{\mu\nu\sigma\rho}$ and $B_l^{\mu\nu}$

The first step in actually performing the computations described in the previous section is the truncation of the Fourier series to a finite number of terms. The numerical approach described above requires truncation of *l* and λ in Eq. (5.2). In this study, $A_{l\lambda}^{\mu\nu\sigma\rho}$ and $B_l^{\mu\nu}$ constants were computed by truncating the series to $\lambda = l = 17$ (a total of 6972 constants for each deformation process) for the two different deformation processes defined in Eq. (5.4). These are denoted as $\left(A_{l\lambda}^{\mu\nu\sigma\rho}\right)_{PSC}$ and $\left(B_l^{\mu\nu}\right)_{PSC}$ for plane strain compression and as $\left(A_{l\lambda}^{\mu\nu\sigma\rho}\right)_{SC}$ and $\left(B_l^{\mu\nu}\right)_{SC}$ for simple compression, respectively. These constants were computed with $\Delta\eta = 0.10$.

If the Fourier representation is extended to an infinite (very large) number of terms, then Bunge and Esling's theory actually expects the representation shown in Eq. (5.2) to be exact. Noting that $A_{l\lambda}^{\mu\nu\sigma\rho}$ and $B_l^{\mu\nu}$ represent rotation tensor fields associated with texture evolution, it should be possible to prove that beyond a certain value of λ and *l*, the constants $A_{l\lambda}^{\mu\nu\sigma\rho}$ and $B_l^{\mu\nu}$ are identically equal to zero (similar analyses have been presented for other tensorial properties [6], but not for these constants).

However, at λ = *l* = *17*, the values of $A_{l\lambda}^{\mu\nu\sigma\rho}$ and $B_l^{\mu\nu}$ are non-zero, indicating that we have not yet reached the required number of terms where the representation in Eq. (5.2) becomes exact. We therefore need to critically evaluate the associated truncation errors.

In an attempt to evaluate the goodness of fit obtained by the χ^2 technique described above, we have also performed the computations by truncating the Fourier series to lower values of λ and *l* (i.e. lower than 17). With each choice of truncation level, we obtained different values of the constants $A_{l\lambda}^{\mu\nu\sigma\rho}$ and $B_l^{\mu\nu}$. One convenient method of evaluating the truncation errors involved is to represent the crystal lattice rotation due to the application of Δη increment of the specified deformation process as a vector in the Fourier space, where the tail of the vector represents the initial crystal orientation and the head of the vector represents the orientation in the deformed state. For example, if the vectors are drawn with $\left(F_l^{\mu\nu}(0)\right)^{(k)}$ as the tail of the vector and with $\left(F_l^{\mu\nu}(\Delta\eta)\right)^{(k)'}$ as the head of the vector for 200 crystal orientations deformed in plane strain compression (rolling), we would obtain a vector flow field that represents the lattice rotation tensor field predicted by Taylor-type model; this flow field is shown in Figure 5 in the first three dimensions of the Fourier space using black colored vectors. Using the values of $\left(A_{l\lambda}^{\mu\nu\sigma\rho}\right)_{PSC}$ and $\left(B_l^{\mu\nu}\right)_{PSC}$ obtained from the χ^2 technique described above, we also computed the Fourier coefficients of the deformed ODF for the same initial orientations using in Eq. (5.2). Such vector fields, recomputed using the obtained values of $\left(A_{l\lambda}^{\mu\nu\sigma\rho}\right)_{PSC}$

and $\left(B_l^{\mu\nu}\right)_{PSC}$ (note that these have different values when truncated to different values of *l* and λ), are also plotted in Figure 5 for the first three dimensions of the Fourier space.

Ideally, when the truncation errors are insignificant, the recomputed vector field would match well with the vector field of the Taylor-type model. A close examination of Figure 5 (a) reveals that the red colored vectors representing the spectral model predictions truncated at $l = 4$ do not match well with the Taylor-type model predictions. This error gets reduced significantly with increasing the number of terms in the Fourier representation as shown in Figure 5(b). In Figure 5 (b), the Taylor-type model predictions almost overlap with the spectral model predictions truncated at $l = 17$ (blue colored vectors).

In order to quantify the goodness of fit obtained by the χ^2 technique, an error estimate has been defined as:

$$Avg.\ Error = \frac{\dfrac{\sum_{k=1}^{1000}\left|\left(\sum_{l=1}^{l_{\max}}\sum_{\mu=1}^{M(l)}\sum_{\nu=1}^{N(l)}\left(E_l^{\mu\nu}\right)^{(k)}\left(E_l^{\mu\nu}\right)^{(k)}\right)^{1/2}\right|}{1000}}{\dfrac{\sum_{k=1}^{1000}\left|\left(\sum_{l=1}^{l_{\max}}\sum_{\mu=1}^{M(l)}\sum_{\nu=1}^{N(l)}\left(D_l^{\mu\nu}\right)^{(k)}\left(D_l^{\mu\nu}\right)^{(k)}\right)^{1/2}\right|}{1000}} \tag{5.6}$$

with

$$\left(E_l^{\mu\nu}\right)^{(k)} = \left(F_l^{\mu\nu}(\Delta\eta)\right)^{(k)'} - \sum_{\lambda=1}^{\lambda_{\max}}\sum_{\sigma=1}^{M(\lambda)}\sum_{\rho=1}^{N(\lambda)} A_{l\lambda}^{\mu\nu\sigma\rho}\left(F_\lambda^{\sigma\rho}(0)\right)^{(k)} - B_l^{\mu\nu} \tag{5.7}$$

$$\left(D_l^{\mu\nu}\right)^{(k)} = \left(F_l^{\mu\nu}(\Delta\eta)\right)^{(k)'} - \left(F_l^{\mu\nu}(0)\right)^{(k)} \tag{5.8}$$

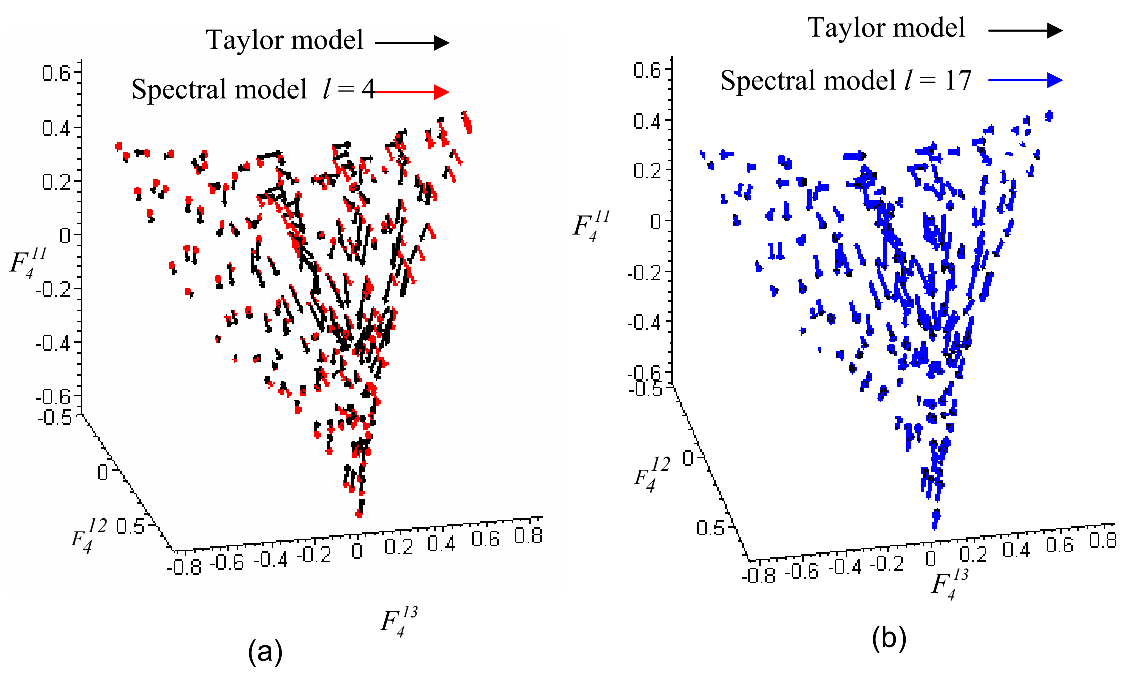


Figure 5. Comparison of the vector flow fields, representing the crystal lattice rotations In the Fourier space, for fcc crystals in an increment of rolling reduction as predicted by the Taylor model and the spectral method. The flow fields are shown only in the three dimensional sub-space corresponding to the first three terms in the Fourier representation of texture. (a) Spectral method truncated at $l = 4$. (b) Spectral method truncated at $l = 17$ terms in the Fourier representation of texture. (a) Spectral method truncated at $l = 4$. (b) Spectral method truncated at $l = 17$.

In the error definition in Eq. (5.6), the numerator represents the average distance between the Taylor-type model predictions and the recomputed values from Eq. (5.2) for the 1000 crystals used in the χ^2 technique, while the denominator represents the average length of the 1000 vectors corresponding to the Taylor-type model predictions. In computing the average error according to Eq. (5.6), we have, in some cases, used different values of $l_{\max}$ and $\lambda_{\max}$ (see equations 5.7 and 5.8). $\lambda_{\max}$ represents the truncation level in the Fourier expansion of the ODF used in Eq. (5.2). The average error computed by Eq. (5.6) is plotted in Figure 6 as a function of the number of terms used in the Fourier expansion for three different cases: (i) $l_{\max} = \lambda_{\max}$ (ii) $l_{\max}= 8$ and (iii) $l_{\max} = 4$. Also shown inset in Figure 6 is the correspondence between the number of independent Fourier coefficients and $\lambda_{\max}$. As can be seen from the figure, there is substantial truncation error even after including large number of terms in the Fourier representation. However, it is also clear the average error continues to drop with increasing number of terms in the Fourier representation. Most importantly, it is quite clear that in order to keep error within acceptable levels, one needs to select $\lambda_{\max}$ significantly larger than $l_{\max}$.

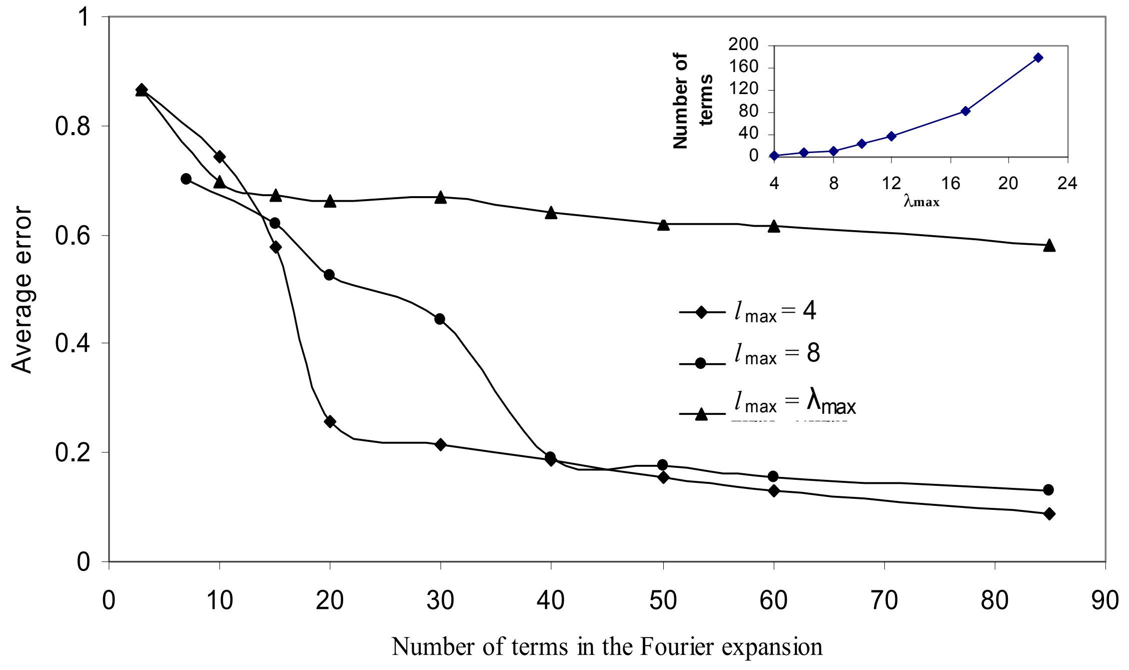
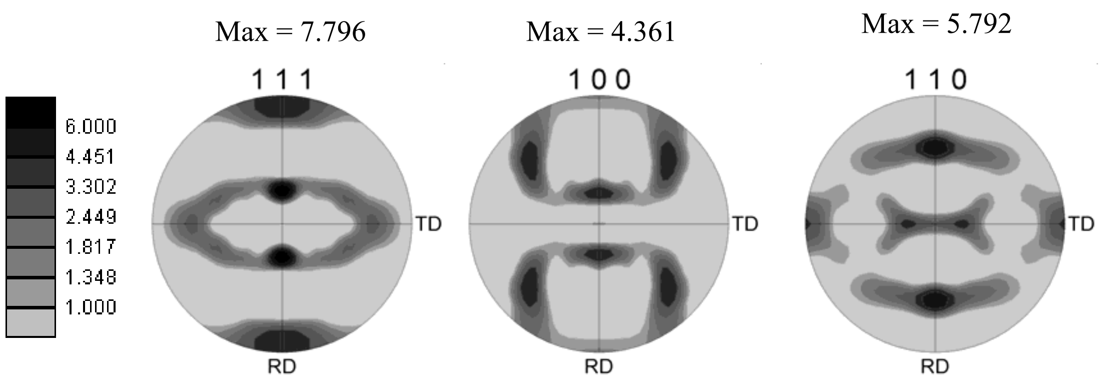


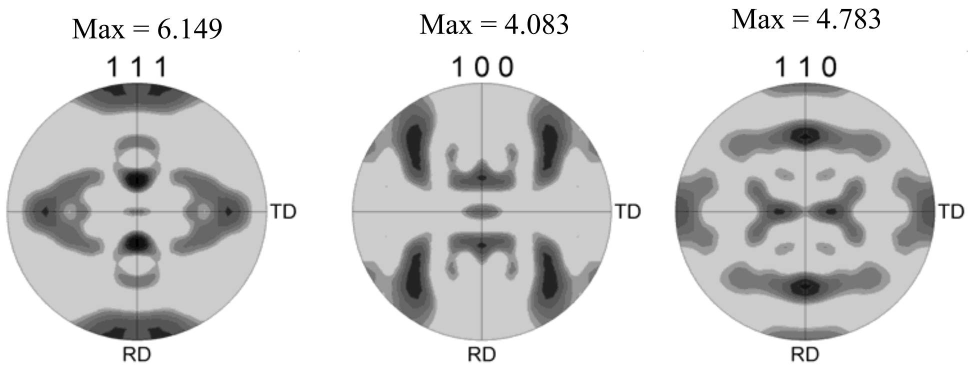
Figure 6. The variation of the average error, for the selected 1000 grain orientations spread in the fundamental zone, with the number of terms included in the Fourier representation. This error reflects the truncation error associated with limiting the Fourier expansion to the selected number of terms.

5.1.3 Predictions of Deformation Textures to Large Strains

The accuracy of Eq. (5.2) to predict deformation textures at large strains at the truncation level described earlier is evaluated. For this purpose, we have selected a random initial texture produced by selection of 1000 grain orientations distributed evenly in the cubic-orthorhombic fundamental zone of the Euler angle space. Fourier coefficients corresponding to the initial texture were computed by averaging the Fourier coefficients of the 1000 selected grain orientations. The Fourier coefficients of the deformed texture in plane strain compression after a true compressive strain of -1.0 were computed by recursive use of Eq. (5.3), ten times, using the $\left(A_{l\lambda}^{\mu\nu\sigma\rho}\right)_{PSC}$ and $\left(B_{l}^{\mu\nu}\right)_{PSC}$ constants computed in the previous section (truncated to $\lambda_{\max} = l_{\max} = 17$). Pole figures corresponding to this final texture were then plotted in Figure 7(a). For comparison, the pole figures plotted using the Taylor-type model predicted grain orientations for the same 1000 initial grain orientations and the same total strain are presented in Figure 7(b). It is seen that the pole figures predicted using $\left(A_{l\lambda}^{\mu\nu\sigma\rho}\right)_{PSC}$ and $\left(B_{l}^{\mu\nu}\right)_{PSC}$ in Eq. (5.2) are in reasonable agreement with the Taylor-type prediction. A similar calculation was also performed for simple compression to a strain of -1.0 using $\left(A_{l\lambda}^{\mu\nu\sigma\rho}\right)_{SC}$ and $\left(B_{l}^{\mu\nu}\right)_{SC}$. The resulting pole figures from Eq. (5.2) are presented in Figure 8(a), while the corresponding Taylor-type prediction is shown in Figure 8(b). The simulation of plane strain compression deformation was repeated with a different initial texture. The deformation texture predicted by the Taylor-type model for simple compression after a true strain of -1.0 as the starting texture for a plane strain compression simulation was used. The calculated pole figures using $\left(A_{l\lambda}^{\mu\nu\sigma\rho}\right)_{PSC}$ and $\left(B_{l}^{\mu\nu}\right)_{PSC}$ in Eq. (5.2) for this fiber-type initial texture and the corresponding Taylor-type prediction are shown in Figures 9(a) and 9(b), respectively. Once again, it is observed that the predictions from Eq. (5.2) are in reasonable agreement with the Taylor-type calculations, indicating that the simple formalism expressed by Eq. (5.2), as proposed by Bunge and Esling, is indeed capable of capturing the main aspects of deformation texture evolution as predicted by the Taylor-type model. It should also be noted that the spectral method described here can be made to better capture the predictions of the Taylor model by further increasing the number of terms in the $A_{l\lambda}^{\mu\nu\sigma\rho}$ and $B_{l}^{\mu\nu}$ matrices.



(a) Taylor-type model



(b) Spectral method

Figure 7. Comparison of the (111), (100), and (110) predicted pole figures in plane strain compression of fcc metals after a compressive true strain of -1.0. The starting texture for These simulations is shown in Figure 3. (a) Taylor model predictions. (b) Spectral method predictions.

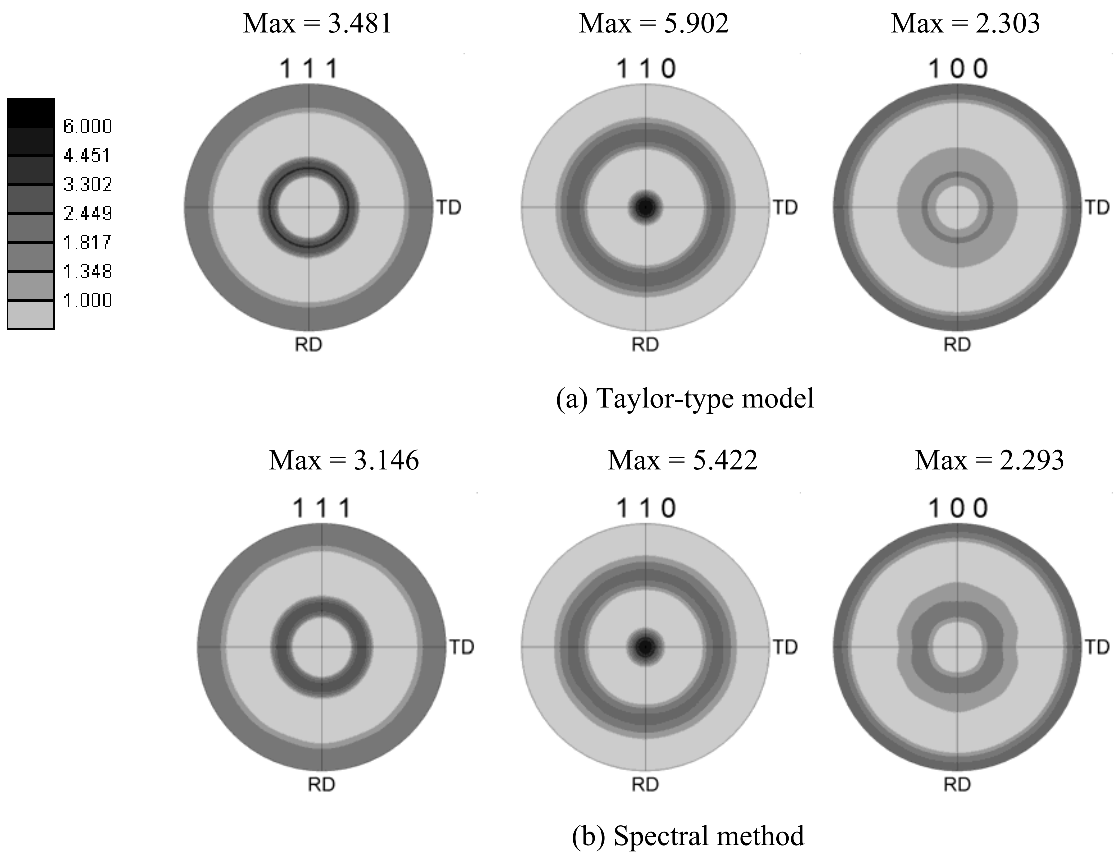


Figure 8. Comparison of the (111), (100), and (110) predicted pole figures in simple compression of fcc metals after a compressive true strain of -1.0. The starting texture for these simulations is shown in Figure 3. (a) Taylor model predictions. (b) Spectral method predictions.

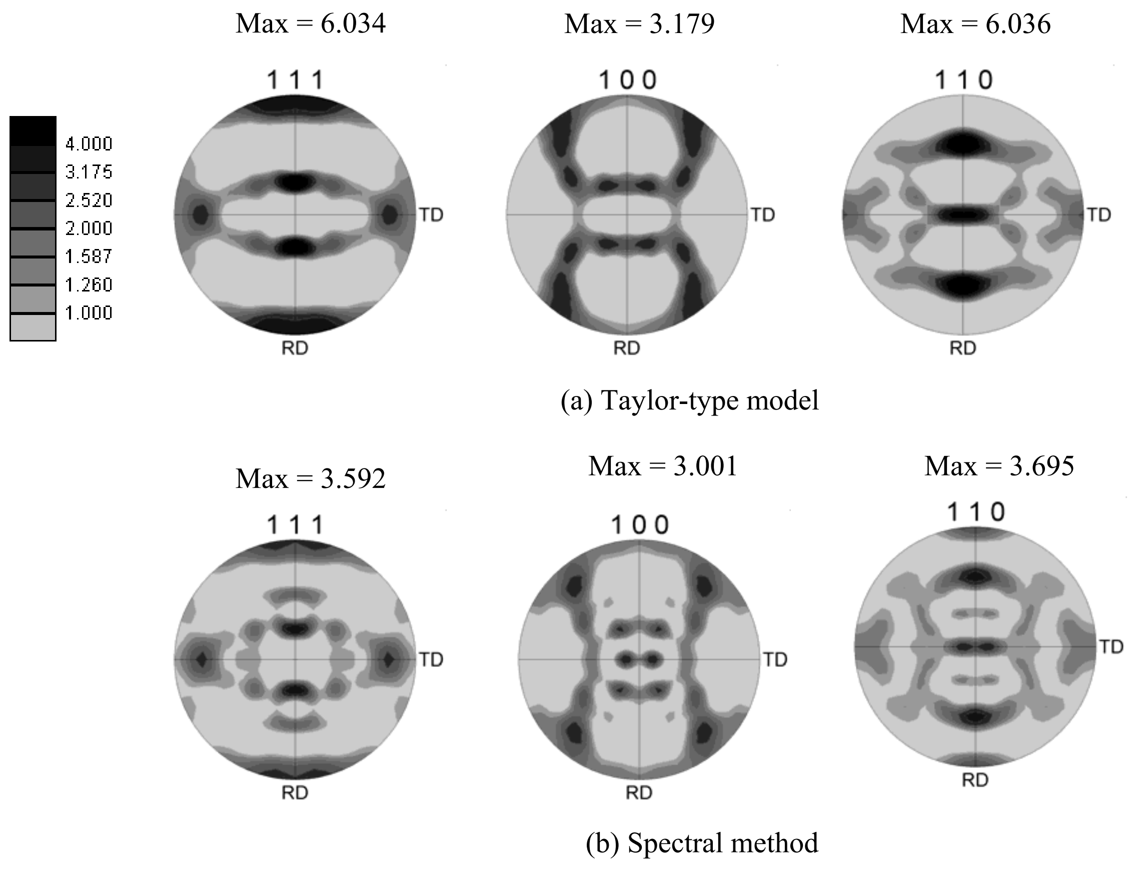


Figure 9. Comparison of the (111), (100), and (110) predicted pole figures in plane strain compression of fcc metals after a compressive true strain of -1.0 with the starting texture shown in Figure 5(a). (a) Taylor model predictions. (b) Spectral method predictions.

From this critical validation of Bunge-Esling approach for crystal plasticity, some of the observations are

i. Though the concept showed great promise, evaluation of the higher-order terms in this specific Fourier representation demands high computational cost.

ii. Although the evaluation of the Fourier coefficients in these representations is a one-time activity, the computational power needed to establish the high-order coefficients was prohibitively high, as it involved the inversion of a very large matrix (in fact the Fourier representations in this work were truncated to *l = 17*, because that was the limit of the computational resources available for this study).

iii. The extension of this method to other deformation processes (e.g. rotations, shear deformation modes) requires the use of cubic-triclinic functions, $\dot{\dot{T}}_l^{\mu\nu}(g)$. This places further constraint in computing the coefficients using the least squares fit method described.

However, Bunge-Esling approach motivated us for a better spectral framework using the GSH basis functions which is described in the next section

5.2 Spectral Crystal Plasticity

A brief review of the crystal plasticity theory presented in chapter 4 suggests that our main focus should be to capture the dependence of stresses, the lattice spins, and the strain hardening rates in individual crystals as a function of their lattice orientation for a specified monotonic deformation path. Specifically, the following representations were formulated:

$$W_{ij}^{*}(g)=\sum_{l=0}^{\infty}\sum_{\mu=1}^{M(l)}\sum_{n=1}^{2l+1} {}_{ij}A_l^{\mu n}\dot{\dot{T}}_l^{\mu n}(g), \tag{5.9}$$

$$\sigma_{ij}(g)=\sum_{l=0}^{\infty}\sum_{\mu=1}^{M(l)}\sum_{n=1}^{2l+1} {}_{ij}S_l^{\mu n}\dot{\dot{T}}_l^{\mu n}(g), \tag{5.10}$$

$$\sum_{\alpha}\left|\dot{\gamma}^{\alpha}\right|(g)=\sum_{l=0}^{\infty}\sum_{\mu=1}^{M(l)}\sum_{n=1}^{2l+1} G_l^{\mu n}\dot{\dot{T}}_l^{\mu n}(g). \tag{5.11}$$

Eqs. (5.9)-(5.11) imply that we are seeking Fourier representations for three independent components of the skew-symmetric $W_{ij}^{*}(g)$, five independent components of the symmetric and deviatoric $\sigma_{ij}(g)$, and for the sum of the absolute values of the slip rates on the different slip systems in the crystal. The variables in Eqs. (5.9)-(5.11) have been selected because they constitute the essential information needed to predict the overall texture evolution and the overall anisotropic stress-strain behavior of cubic polycrystals. Only the sum of the absolute slip rates in each crystal is being represented in this spectral formulation, because this is the only information that is needed for capturing the slip hardening description in Eq. (4.5). If latent hardening is introduced into Eq. (4.5), then it would be necessary to track the slip rates on the different slip systems individually. In order to extend the formalism in Eqs. (5.9)-(5.11) to any arbitrary deformation mode, generalized deformation path should be considered. The most generic isochoric deformation path to be imposed on a polycrystalline metal can be expressed in the principal frame of the stretching tensor as [56]

$$L = \frac{\dot{\varepsilon}}{\dot{\varepsilon}_o} D_o + W, \quad D_o = \begin{bmatrix} \frac{cos\,\theta}{\sqrt{2+sin\,2\theta}}\dot{\varepsilon}_o & 0 & 0 \\ 0 & \frac{sin\,\theta}{\sqrt{2+sin\,2\theta}}\dot{\varepsilon}_o & 0 \\ 0 & 0 & \frac{-(cos\,\theta + sin\,\theta)}{\sqrt{2+sin\,2\theta}}\dot{\varepsilon}_o \end{bmatrix}, \quad (5.12)$$

where $\dot{\varepsilon}_o$ is a reference value of strain rate and the range of angle θ is $[0,2\pi)$, and **W** is an arbitrary superimposed spin tensor (describing an arbitrary rotation rate). The introduction of the reference strain rate and the superimposed spin in the deformation path in Eq. (5.12) requires us to modify Eqs. (5.9) - (5.11) as

$$W_{ij}^{*}(g,\theta) = \frac{\dot{\varepsilon}}{\dot{\varepsilon}_o} \sum_{l=0}^{\infty} \sum_{\mu=1}^{M(l)} \sum_{\nu=1}^{2l+1} \sum_{k=-\infty}^{\infty} {}_{ij}A_l^{\mu\nu k} \dot{T}_l^{\mu\nu}(g) e^{ik\theta} + W_{ij} \quad (5.13)$$

$$\sigma_{ij}(g,\theta) = s \left| \frac{\dot{\varepsilon}}{\dot{\varepsilon}_o} \right|^m \operatorname{sgn}\left(\frac{\dot{\varepsilon}}{\dot{\varepsilon}_o} \right) \sum_{l=0}^{\infty} \sum_{\mu=1}^{M(l)} \sum_{\nu=1}^{2l+1} \sum_{k=-\infty}^{\infty} {}_{ij}S_l^{\mu\nu k} \dot{T}_l^{\mu\nu}(g) e^{ik\theta} \quad (5.14)$$

$$\sum_{\alpha}\left|\dot{\gamma}^{\alpha}\right|(g,\theta)=\left|\frac{\dot{\varepsilon}}{\dot{\varepsilon}_o}\right|\sum_{l=0}^{\infty}\sum_{\mu=1}^{M(l)}\sum_{\nu=1}^{2l+1}\sum_{k=-\infty}^{\infty}G_l^{\mu\nu k}\dot{T}_l^{\mu\nu}(g)e^{ik\theta} \tag{5.15}$$

The Fourier coefficients in Eqs. (5.13)-(5.15) can be expressed as

$${}_{ij}A_l^{\mu nk}=\frac{(2l+1)}{2\pi}\int_0^{2\pi}\oint W_{ij}^{*}(g,\theta)\overline{\dot{T}_l^{\mu n}}(g)e^{-ik\theta}\,dg\,d\theta \tag{5.16}$$

$${}_{ij}S_l^{\mu nk}=\frac{(2l+1)}{2\pi}\int_0^{2\pi}\oint \sigma_{ij}(g,\theta)\overline{\dot{T}_l^{\mu n}}(g)e^{-ik\theta}\,dg\,d\theta \tag{5.17}$$

$$G_l^{\mu nk}=\frac{(2l+1)}{2\pi}\int_0^{2\pi}\oint \sum_{\alpha}\left|\dot{\gamma}^{\alpha}\right|(g,\theta)\overline{\dot{T}_l^{\mu n}}(g)e^{-ik\theta}\,dg\,d\theta \tag{5.18}$$

The coefficients in Eqs. (5.16) - (5.18) can be evaluated numerically. For this purpose, one needs to select a distribution of grain orientations in the fundamental zone of the cubic-triclinic orientation space [6] and compute the values of W_{ij}^{*}, σ_{ij}, and $\sum_{\alpha}\left|\dot{\gamma}^{\alpha}\right|$ for each of the selected orientation using the crystal plasticity models described in chapter 4.

In this study , W_{ij}^{*}, σ_{ij}, and $\sum_{\alpha}\left|\dot{\gamma}^{\alpha}\right|$ were computed for each of the selected orientations in the fundamental zone using the Taylor-type model described for a small strain step corresponding to $\varepsilon=-0.02$ for the general deformation modes described by the velocity gradient tensor in Eq. (5.12). This information can then be used in Eqs. (5.16) - (5.18) using an appropriate numerical integration scheme (e.g. the Simpson method [24]) to obtain the desired Fourier coefficients. After the Fourier coefficients are established, Eqs. (5.13)-(5.15) can be used in a recursive manner to simulate large plastic strains on the polycrystals.

5.2.1 Results and Discussion

Compared to that of Bunge-Esling approach, the merits of spectral crystal plasticity are

i. The Fourier coefficients can be computed to very high orders in the series expansion as they do not require a least squares fit and the associated inversion of large matrices. This, in turn, has allowed us to address cubic-triclinic symmetry needed in representing the most general rotation fields for cubic crystals, whereas the computational demands in our previous model restricted us to the consideration of only the cubic-orthorhombic symmetry.

ii. The estimation of the Fourier coefficients in this new formulation is completely uncoupled, i.e. the Fourier coefficients can be obtained individually by numerical integration over the orientation space. This means that one can augment the number of terms in the series expansions depending on the accuracy needed for a particular application.

iii. The present formulation allows for Fourier representations of local stresses (all tensorial components) and strain hardening rates in the constituent crystals in addition to their lattice spins. Consequently, it would be possible to predict both the stress-strain curves and the texture evolution using the new formulation.

In order to evaluate the accuracy of the Fourier representations obtained by calibration to the Taylor-type model in Eqs. (5.13)-(5.15), we simulated couple of deformation modes.

Plane Strain Compression

Simulated plane strain compression to $\varepsilon = -1.0$ for two different initial textures: (1) a random initial texture, and (2) a <110> fiber texture obtained by simulating simple compression to a strain of -1.0 on a initially random texture using the Taylor-type model. Deformed textures and the anisotropic stress-strain curves were computed using both the Taylor-type model as well as the spectral representations. In the spectral method, we applied Eqs. (5.13)-(5.15), recursively fifty times, each time for a strain step of -0.02. In each strain step, the grain orientations are updated and the new orientations are used in the next strain step. The predicted textures from both the Taylor-type models and the spectral methods described here are shown in Figs. 10 and 11, while the corresponding predictions of the stress-strain responses are shown in Fig. 12. It is seen that the spectral methods accurately reproduced all of the important aspects of the Taylor-type model predictions.

Simple Shear

As the next case study we simulated simple shear deformation. Deformation textures and the stress-strain curves in simple shear after a strain of $\gamma = 1.0$ were predicted from both the Taylor-type model and the spectral method and are shown in Figs. 13 and 14. It is seen once again that the spectral method reproduces accurately the results of the Taylor-type models.

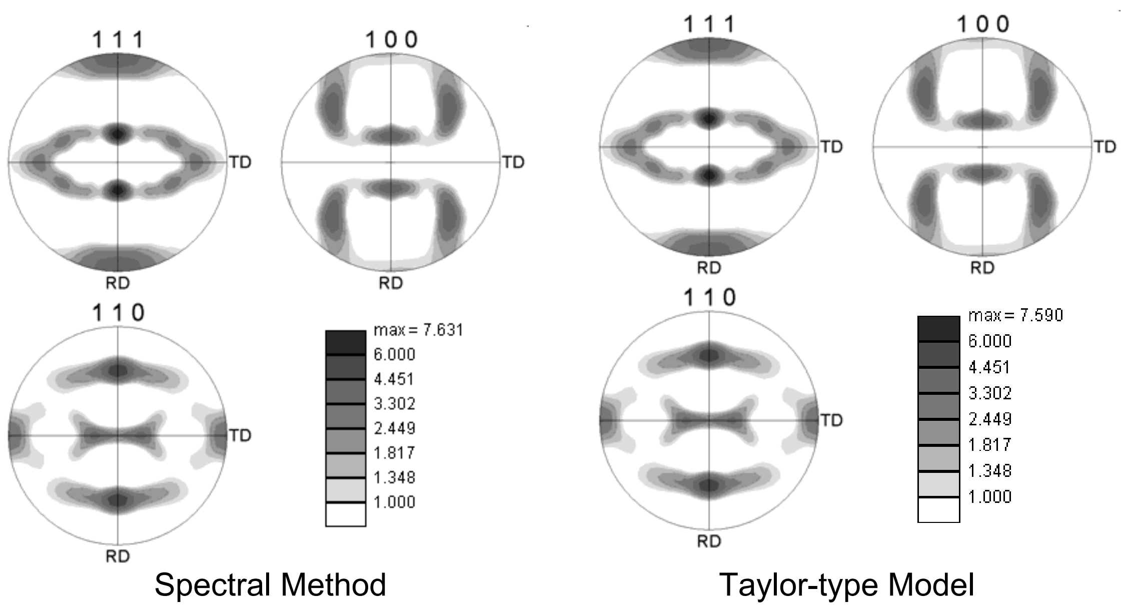


Figure 10. Comparison of the predicted textures in plane strain compression of fcc metals after a compressive true strain of -1.0. The starting texture for these simulations is a random texture.

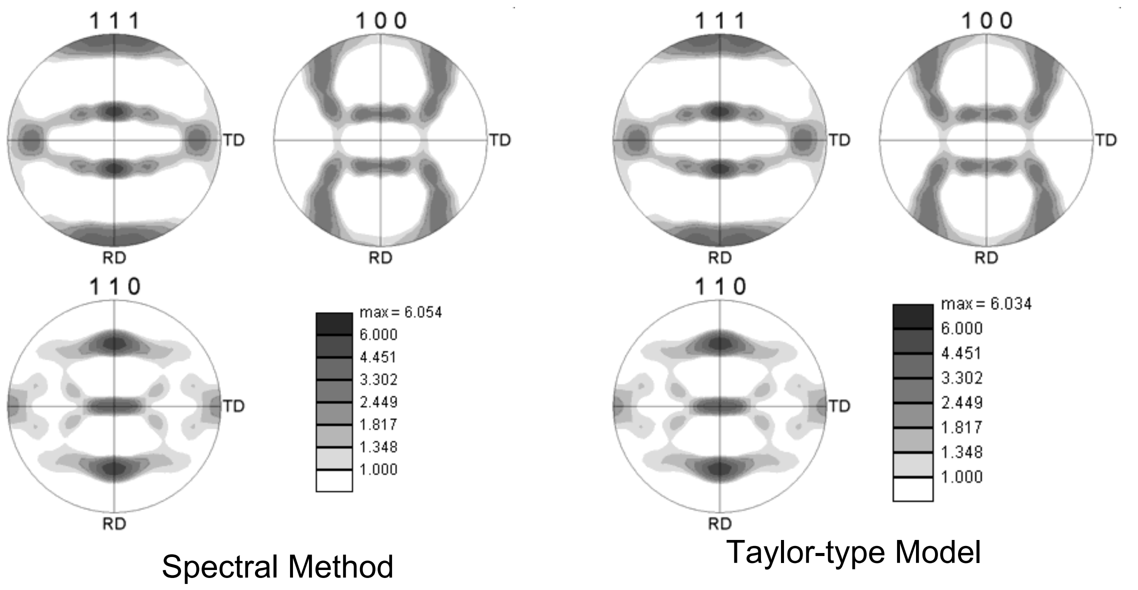
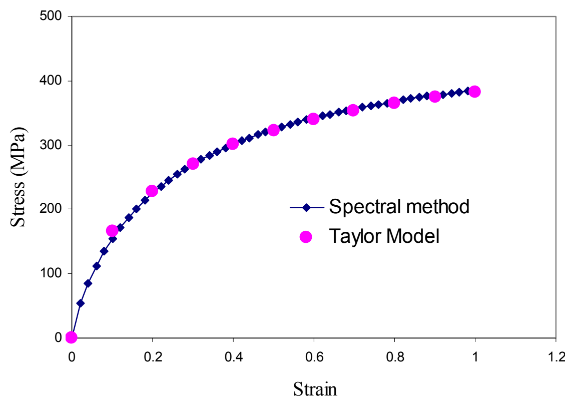
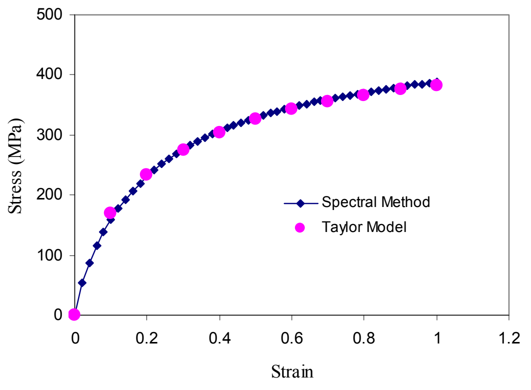


Figure 11. Comparison of the predicted textures in plane strain compression of fcc metals after a compressive true strain of -1.0. The starting texture for these simulations is a <110> fiber texture obtained by simulating simple compression to a strain of -1.0 on an initially random texture using the Taylor-type model.



(a) Random initial texture.



(b) Initial texture is <110> fiber texture

Figure 12. Stress-strain responses in plane strain compression. (a) Random initial texture. (b) Initial texture is <110> fiber texture.

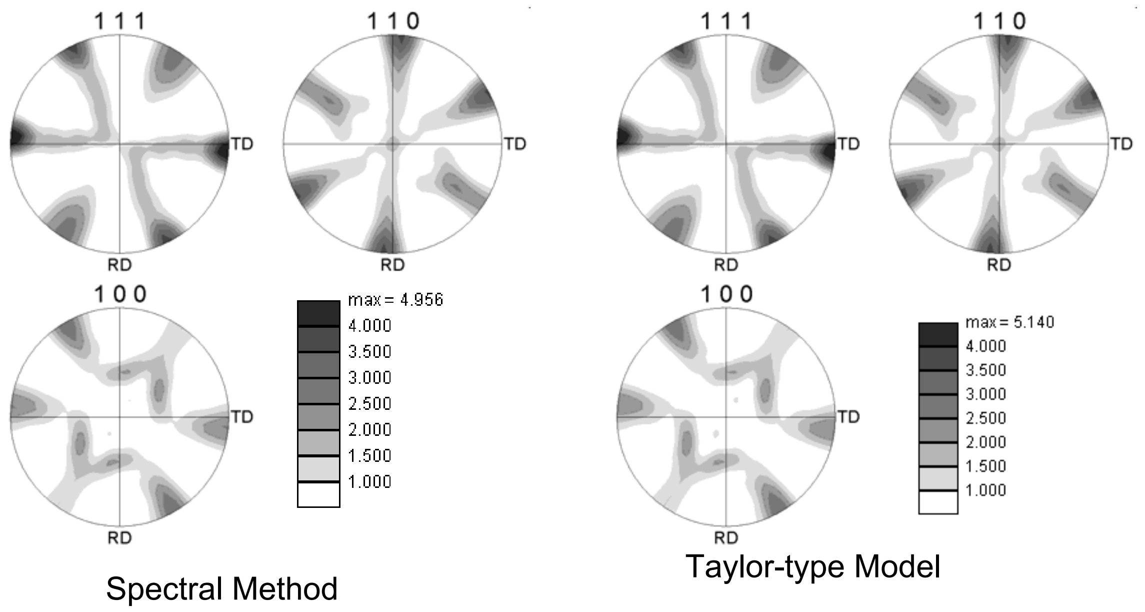


Figure 13. Comparison of the predicted odfs in simple shear of fcc metals after a shear strain of 1.0.

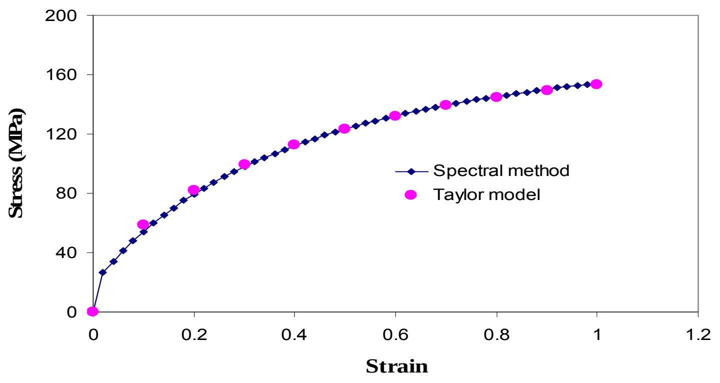


Figure 14. Stress-strain responses in simple shear of fcc metals subjected to a shear strain of 1.0.

Deformation Path Change

As a final case study, investigated texture predictions and stress-strain response in a change of deformation mode. Random initial texture is subjected to a strain of $\varepsilon = -1.0$ in plane strain compression and then to a strain of $\gamma = 1.0$ in simple shear. Corresponding texture predictions and stress-strain response are shown in Figs. 15 and 16. Here too the predictions from the spectral method and the Taylor-type model are in very good agreement.

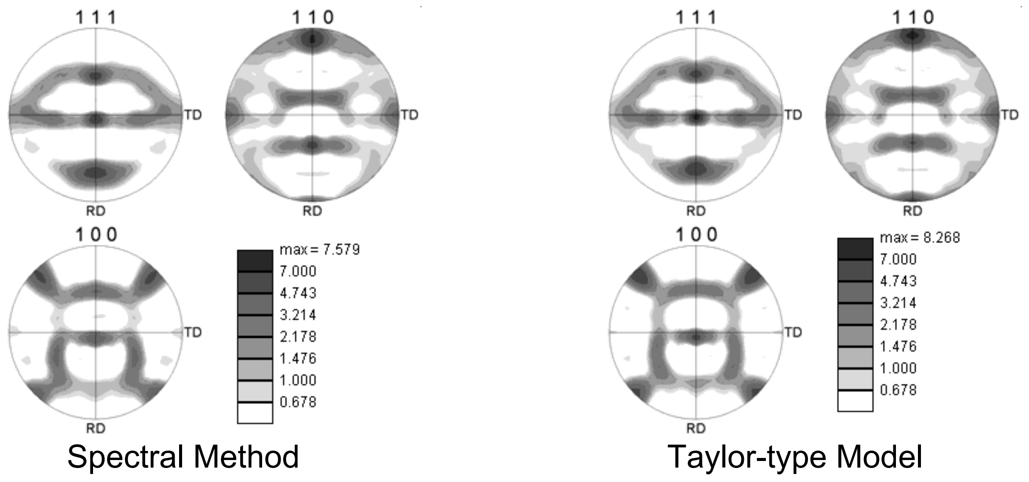


Figure 15. Comparison of the predicted textures in fcc metals subjected to plane strain compression and then to simple shear deformation. The starting texture for these simulations is a random texture.

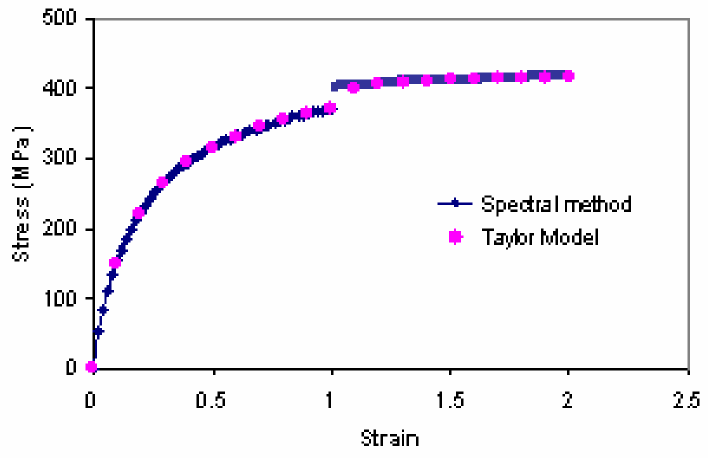


Figure 16. Stress-strain responses in change of deformation mode. The initial random texture is subjected to plane strain compression and then to simple shear.

From these case studies, it is demonstrated that the spectral methods do accurately reproduce almost all of the salient features of the predictions from the Taylor-type model and this is quite remarkable. However, the computational time that we expected to reduce with this spectral crystal plasticity was not dramatical (it was not orders of magnitude less compared to that of conventional crystal plasticity models). With this work, we found out that the computation of the generalized spherical harmonics itself was the speed limiting factor. As described in Chapter 2 GSH functions involves computation of Legendre functions and these are quite complex mathematically. One of the main advantages of using the generalized spherical harmonics was that they are already symmetrized to reflect the appropriate crystal and sample symmetries, and therefore would provide a compact representation of the orientation dependence of any variable of interest in the crystal plasticity computations crystal plasticity computations[6]. Following a similar approach as that of the spectral crystal plasticity presented in this section, it was very recently discovered that it is much more efficient computationally to simply store all of the important variables of interest on a uniform grid in the orientation space and subsequently employ a local spectral interpolation using Discrete Fast Fourier Transform (DFFT) methods to recover the values of any of these variables for any orientation and deformation mode of interest [57]. This an ongoing work in our research group and details of this new approach are described and validated in this paper [58] through a few example case studies involving crystal plasticity calculations.

CHAPTER 6: INCORPORATION OF SPECTRAL CRYSTAL PLASTICITY INTO FORGE3

The objective in this part of the thesis is to incorporate the newly developed spectral crystal plasticity framework described in chapter 5 into commercially available finite element codes to simulate deformation processing operations.

Various concepts exist to incorporate texture information into finite element models to simulate large deformation of polycrystalline solids, particularly, in metal forming including bulk forming and sheet forming operations. The initial material anisotropy existing before sheet deformation can be incorporated through an anisotropic yield surface function into the finite element codes. The anisotropic yield surface models are empirical and phenomenological and the widely used models are the equations of Hill from 1948 [59] and 1990 [60], Hosford [8], Barlat [61], or Barlat and Lian [62] to name but a few important ones. The advantage of phenomenological theories for mechanical anisotropy predictions are relatively short calculation times, when implemented into finite element models. The main disadvantage of phenomenological models is that they do not consider that the inherited sheet starting textures may evolve further in the course of sheet forming. This means that reliable anisotropy simulations should incorporate the starting texture as well the gradual update of that texture during deformation processes.

Currently, there are two types of crystal plasticity finite element applications: first is the prediction of evolving microstructure; second is the determination of properties associated with the evolving microstructure. Some simulations involve only one of the two, while others couple the two together. The basic approach for studying metal deformations using the finite element method as a numerical tool is that a polycrystalline aggregate containing many grains is discretized using the finite element technique. The microstructural state of the material is characterized by the orientation of the crystal lattice and the constitutive response is evaluated at specific locations in each element called integration points. In general, crystal plasticity theories are incorporated into finite element codes through use of a user-defined subroutine for material behavior called UMAT. The UMAT keeps track of the necessary solution dependent internal state variables at each of the integration points in the finite element mesh, and updates them during an imposed deformation increment. In addition, some of the finite element codes require a computation of the Jacobian matrix at each integration point in the mesh in order to compute a better guess of the overall non-uniform deformation field in its iterative attempts to satisfy the principle of virtual work during an imposed increment of deformation[27].

In this first foray in coupling finite element codes and the database approach for crystal plasticity, focus was restricted to predict the evolving microstructure during deformation. The concomitant anisotropic stress-strain response is not considered. Because of the speed limiting factor of the GSH basis functions as remarked in chapter 5, spectral databases using the Discrete Fast Fourier Transform (DFFT) methods [57, 58] are used in coupling into finite element codes.

The finite element S/W package that is used in this work is called FORGE3 which is commercially available and specifically developed for various thermo-mechanical metal forming simulations. FORGE3 provides an interface where by the user may write his or her own constitutive model in a subroutine in a very general way. Integrating spectral crystal plasticity DFFT databases and their associated computations (programmed in

MATLAB S/W package) and FORGE3 codes (programmed in FORTRAN S/W language) required to develop special S/W protocols. The corresponding S/W codes are presented in appendix.

In order to validate the coupling of spectral database and FORGE3 codes, texture evolution during simple compression deformation process was investigated. For this purpose a 3D finite element model comprising 12 tetrahedron elements (5 nodes for each tetrahedron element and so a total of 60 integration points) was developed. A random initial texture consisting of 1000 crystals was assigned to each integration point and the model was subjected to 75% deformation. Fiber textures that developed during simple compression deformation process with varying intensities at various stages of deformation are shown in Fig. 17 along with the deformed geometry. This coupled simulation took approximately 6 minutes on a regular desktop PC, where as the same simulation would take more than an hour with conventional crystal plasticity models. It should be also be noted that many other strategies are currently being explored in our research group to further speed up the calculations involving spectral databases. The details of those strategies are beyond the scope of this present work.

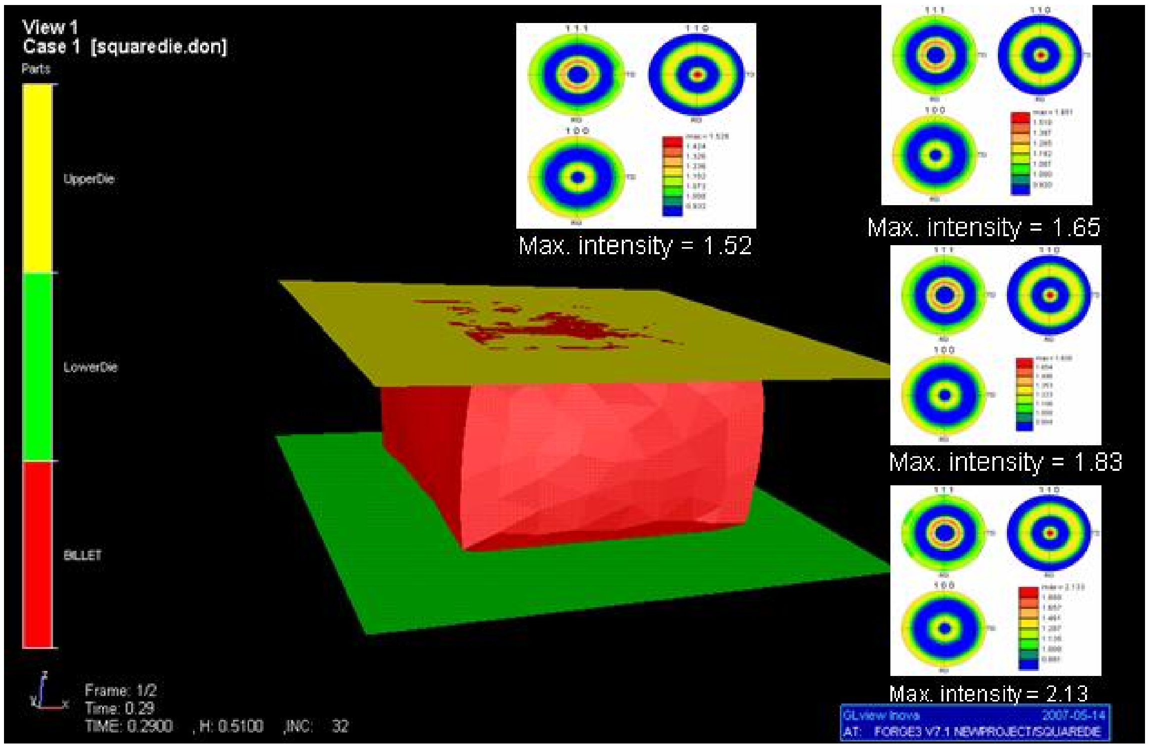


Figure 17. Deformed geometry of a 3D finite element model subjected to simple compression. Crystallographic textures that are predicted at various stages during the deformation process are also shown.

CHAPTER 7: PROCESS DESIGN

The major goal of process design is to develop rigorous mathematical procedures that can identify one or more processing paths that are predicted to produce either an element of the class of optimal microstructures, or a microstructure (presumably suboptimal) that is close to the optimal set, and one that could be realized within reasonable cost by a small set of available manufacturing routes. There are no rigorous solution methodologies to the process design problem yet. One strategy that was investigated in this work is schematically shown in Figure 18. Before describing the strategy for process design, a brief mention of texture hull is necessary. As remarked earlier in chapter 2, Fourier description of ODF allows the visualization of ODF as a single point in an infinite dimensional Fourier space (coordinates given by $F_l^{\mu\nu}$). The set of all such points, corresponding to the complete set of all physically realizable ODFs, is called the texture hull in the MSD framework [1, 2]. The hull for cubic-orthorhombic textures is depicted in the first three dimensional Fourier subspace in Fig. 18. It should be recognized that any physically realizable texture has to have a representation inside the depicted hull.

Note also that the texture hulls are compact and convex in any of their subspaces [5] .

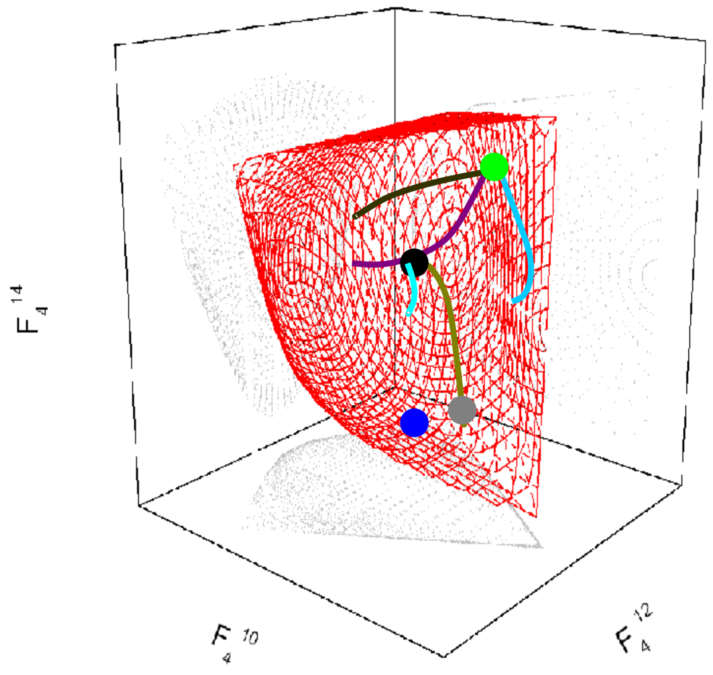


Figure 18. A schematic description of a strategy to find process design solutions. The circles indicate specific textures and the lines describe streamlines (or paths) of texture evolution for a selected processing method.

The proposed methodology for process design is as follows. Let the green circle in the far right in the convex hull represent the microstructure of the as-received material and the blue circle on the far left represent the location in Fourier space of a class of microstructures with desired combination of properties. The problem at hand is to find an overall processing route that will transform the given initial microstructure (green circle in Fig 18) to an element of the family of desired microstructure(s) (blue circle). The processing route has to be made up of segments chosen from a set of available manufacturing routes. If it is not possible to reach the desired microstructure(s) using the available manufacturing routes, then process design aims to get as close as possible to the desired microstructure(s). For each available manufacturing process, one could envision developing the spectral representations of the type described in Chapter 5. Figure 18 shows schematically three different microstructure evolution paths starting from the green circle, each corresponding to an available manufacturing route. These paths can be computed very efficiently for any given location in the microstructure hull using the spectral databases described in Chapter 5.

7.1 Case Study

As a case study, an orthotropic thin plate with a central circular hole and subjected to in-plane tensile loads is considered. The design objective is to maximize the load carrying capacity of the plate, without allowing plastic deformation. An optimum texture was identified using the MSD framework [2, 63]. The goal is now to obtain a processing solution to obtain the desired texture starting from a random initial texture. The processing routes considered here included only those room temperature deformation processes that preserve the orthorhombic sample symmetry. This limited set of processing paths can be defined through the following velocity gradient tensor.

$$L = \begin{bmatrix} \alpha \dot{\varepsilon} & 0 & 0 \\ 0 & \beta \dot{\varepsilon} & 0 \\ 0 & 0 & -(\alpha + \beta)\dot{\varepsilon} \end{bmatrix} \quad \alpha, \beta \in [-1, 1] \tag{7.1}$$

Figure 19 delineates the region corresponding to all physically realizable textures, in the first three component subspace of the Fourier space, as a gray wire-framed convex hull [4]. However, not all textures in this convex hull can be realized by the selected set of processing paths; the set of textures that can be produced is depicted as a black wire-framed region in Figure 19. The proposed methodology allows us to identify the complete set of textures that can be produced, starting from a given initial texture, by an arbitrary combination of selected processing techniques. It is further possible to design a processing route for any selected target texture that lies in this subset of realizable textures (the black wire-framed subspace in this case study).

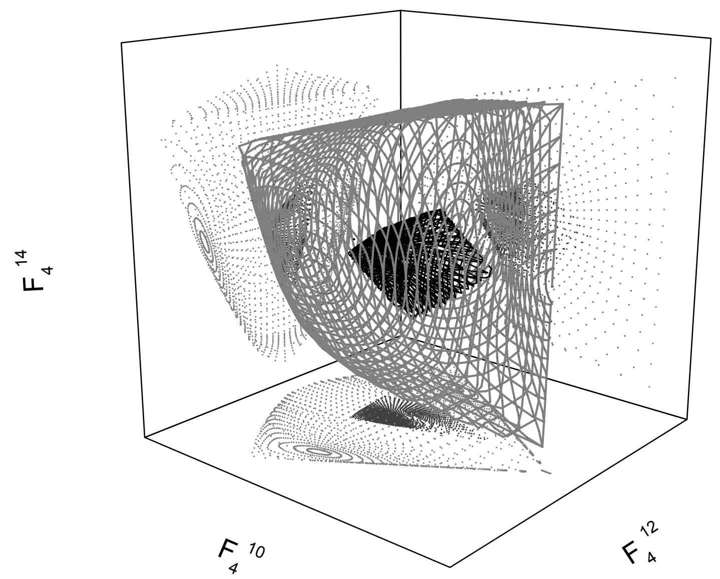


Figure 19. The gray wire-framed convex hull denotes the set of all possible textures, while the black subspace denotes the set of textures that can be produced by a combination of selected processing routes starting with a random initial texture.

The above case study reveals that the none of the elements of the previously identified class of textures that were deemed best for the thin orthotropic plate with a circular hole can be produced starting with an initial random texture. Therefore, the MSD optimization computations were repeated a second time, and this time the choice of Fourier coefficients in the MSD optimization computations was restricted to the inner region in Fig. 19, because we know we can realize these textures by deformation processing (starting with a random initial texture). In this manner, a target texture was identified as the texture that would yield the best possible performance for the selected design case study that is also easily producible using deformation processing steps. The texture identified is shown in Fig. 20 as a (111) pole figure and Taylor-type models predict that this texture can be produced in a two step rolling process: a 50% rolling of the plate followed by a 20% rolling in a direction perpendicular to the original rolling direction.

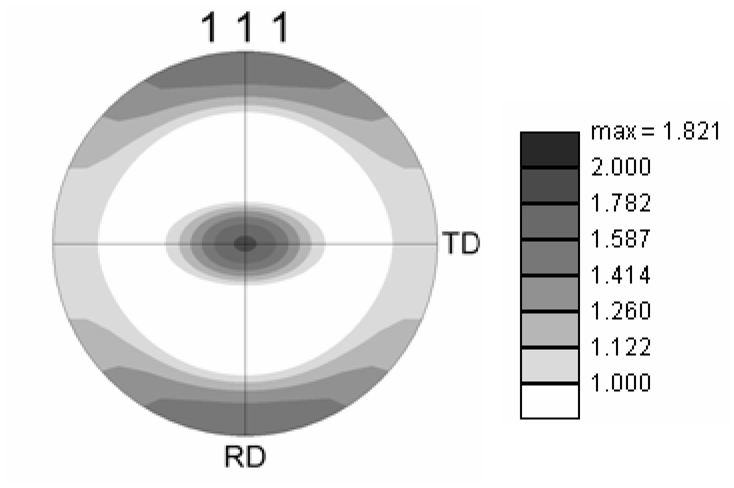


Figure 20. (111) pole figure corresponding to the Fourier coefficients of the ODF identified as the optimized texture in the MSD framework for the present case study, while being producible using deformation processing steps on a material with an initial random texture.

CHAPTER 8: CONCLUSIONS AND FUTURE WORK

The following conclusions can be drawn from three different sections of this work.

Microstructure-Property Linkage

- A new spectral framework has been established to develop efficient scale-bridging laws that link the quantitative statistical descriptors of the microstructure at the lower length scale to the effective properties exhibited by the microstructure at the higher length scale.
- The new framework is built around a localization relationship that provides information on the spatial variation of important local variables at the microstructural length scales. These localization relationships can prove very valuable in formulating microstructure based macroscale property/performance models.
- The proposed framework was successfully applied to perfectly disordered random textures in copper polycrystals. The spectral linkages developed provided excellent predictions of local stresses in individual crystals as well as the components of the effective stiffness tensor.

Spectral Crystal Plasticity

- A new spectral framework is presented to capture efficiently the predictions for the stresses, the lattice spins, and the strain hardening rates in individual crystals from the currently used crystal plasticity models as a function of the crystal's lattice orientation.
- It is particularly noteworthy that although the spectral methods were calibrated by simulating the Taylor-type model to only a small strain increment of 2%, the

recursive use of this calibration yields excellent predictions up to very large strain levels.

- Incorporated the spectral crystal plasticity databases into FORGE3 finite element tool for texture evolution and validated in simple compression deformation process.

Process Design

- The proposed process design methodology allows us to identify the complete set of textures that can be produced, starting from a given initial texture, by an arbitrary combination of selected processing techniques.
- It is also possible to design a processing route for any selected target texture that lies in the subset of realizable textures

Future Work

- Spectral crystal plasticity framework can be extended to other class of metals, e.g. hcp metals.
- At present, only texture prediction calculations were performed in coupling the FORGE3 finite element tool and spectral databases. This can be extended to predict the anisotropic stress-strain behavior along with the texture evolution calculations by including the computation of the Jacobian matrix.
- In the process design, selected manufacturing processes were limited to those that retain cubic-orthorhombic symmetry. The methodology can be extended to include all possible deformation processes.

List of References

1. Adams BL, Henrie A, Henrie B, Lyons M, Kalidindi SR, Garmestani H, Microstructure-sensitive design of a compliant beam. Journal of the Mechanics and Physics of Solids, 2001. 49(8): p. 1639-1663.

2. Kalidindi SR, Houskamp JR, Lyons M, Adams BL, Microstructure sensitive design of an orthotropic plate subjected to tensile load. International Journal of Plasticity, 2004. 20(8-9): p. 1561-1575.

3. Adams BL, Lyon M, and Henrie B, Microstructures by design: linear problems in elastic-plastic design. International Journal of Plasticity, 2004. 20(8-9): p. 1577-1602.

4. Adams BL, Garmestani H, Saheli G, Microstructure design of a two phase composite using two-point correlation functions. Journal of Computer-Aided Materials Design, 2004. 11: p. 103–115.

5. Lyon M, Adams BL, Gradient-based non-linear microstructure design. Journal of the Mechanics and Physics of Solids, 2004. 52: p. 2569-2586.

6. Bunge H-J, Texture analysis in materials science. Mathematical Methods. 1993, Göttingen: Cuvillier Verlag.

7. Raabe D, Wang Y, Roters F, Crystal plasticity simulation study on the influence of texture on earing in steel Computational Materials Science 2005. - 34(- 3): p. -234.

8. Hosford WF, Caddell RM, Metal Forming Mechanics and Metallurgy. Prentice-Hall, Inc., 1993.

9. Gan W, Babu SS, Kapustka N, Wagoner RH, Microstructural Effects on the Springback of Advanced High-Strength Steel. Metallurgical and Materials Transactions, 2006. 37A.

10. Bunge H, Texture Analysis in Materials Science. Butterworths, 1982.

11. Randle V, Engler O, Introduction to Texture Analysis. Macrotexture, Microstructure & Orientation Mapping. 2000: Gordon and Breach Science Publishers.

12. Adams BL, Kalidindi SR, Fullwood D, Microstructure Sensitive Design for Performance Optimization. BYU Academic Publishing, 2006.

13. ISO/IEC IS 10918-1 | ITU-T Recommendation T.81.

14. http://en.wikipedia.org/wiki/MP3.

15. Arfken G, Mathematical Methods for Physicists. Vol. 3rd ed. 1985, Orlando, FL:: Academic Press. 481-485.

16. Proust G, Kalidindi SR, Procedures for construction of anisotropic elastic-plastic property closures for face-centered polycrystals using first order bounding relations. Journal of the Mechanics and Physics of Solids, 2005. in press.

17. Kalidindi SR, Houskamp JR, Proust G, Coupling of Microstructure Sensitive Design with Finite Element Codes: Application to Design of an Orthotropic Plate with a Central Circular Hole Loaded in Tension. Proceedings of DETC2004:ASME 2004 International Design Engineering Technical Conferences And The Computers And Information In Engineering Conference (DETC'04), 2004.

18. Proust G, Kalidindi SR, Procedures for construction of anisotropic elastic-plastic property closures for face-centered cubic polycrystals using first-order bounding relations. Journal of the Mechanics and Physics of Solids, 2006. 54(8): p. 1744-1762.

19. Beran MJ, Mason TA, Adams BL, Olsen T, Bounding elastic constants of an orthotropic polycrystal using measurements of the microstructure. Journal of the Mechanics and Physics of Solids, 1996. 44(9): p. 1543-1563.

20. Kröner E, Bounds for effective elastic moduli of disordered materials. J. Mech. Phys. Solids, 1977. 25: p. 137-155.

21. Milton GW, The Theory of Composites Cambridge Monographs on Applied and Computational Mathematics, 2001.

22. Torquato S, Random Heterogeneous Materials. 2002, New York: Springer-Verlag.

23. Press WH, Teukolsky SA, Vetterling WT, Flannery B, Numerical Recipes in C++. 2002.

24. ABAQUS, Reference Manuals. 2006, Hibbit, Karlsson, and Sorensen, Inc.

25. Meyers MA, Chawla KK, Mechanical Behavior of Materials. Prentice Hall, 1999.

26. Duvvuru HK, Wu X, Kalidindi SR, Calibration of Elastic Localization Tensors to Finite Element Models: Application to Cubic Polycrystals. Computational Materials Science, 2007. In press.

27. Kalidindi SR, Bronkhorst CA, Anand L, Crystallographic Texture Evolution in Bulk Deformation Processing of Fcc Metals. Journal of the Mechanics and Physics of Solids, 1992. 40(3): p. 537-569.

28. Asaro RJ, Micromechanics of crystals and polycrystals. Advances in Applied Mechanics, 1983. 23: p. 1-115.

29. Asaro RJ, Needleman A, Texture development and strain hardening in rate dependent polycrystals. Acta Metallurgica et Materialia, 1985. 33(6): p. 923-953.

30. Sachs G, Zur Ableitung einer Fliebedingung. Z. Ver. Deu. Ing.,, 1928. 72(22).

31. Taylor GI, Plastic strain in metals. Journal of the Institute of Metals, 1938. 62: p. 307-324.

32. Kocks UF, Tome CN, Wenk H.-R, Texture and Anisotropy. 1998, Cambridge: Cambridge University Press.

33. Bishop JFW, Hill R, A theory of the plastic distortion of a polycrystalline aggregate under combined stresses. Philosophical magazine, 1951. 42(7).

34. Hutchinson JW, Proceedings of Royal Society of London A., 1970. 319(247).

35. Sarma GB, Dawson PR, Texture predictions using a polycrystal plasticity model incorporating neighbor interactions. International Journal of Plasticity, 1996. 12(8): p. 1023-1054.

36. Sarma GB, Dawson PR, Effects of interactions among crystals on the inhomogeneous deformations of polycrystals Acta Materialia 1996. - 44(- 5): p. - 1953.

37. Beaudoin AJ, Dawson PR, Mathur KK, Kocks UF, A hybrid finite element formulation for polycrystal plasticity with consideration of macrostructural and microstructural linking. International Journal of Plasticity, 1995. 11(5): p. 501-521.

38. Raabe D, Roters F, Using texture components in crystal plasticity finite element simulations. International Journal of Plasticity, 2004. 20(3): p. 339-361.

39. Raabe D, Sachtleber M, Zhao Z, Roters F, Zaefferer S, Micromechanical and macromechanical effects in grain scale polycrystal plasticity experimentation and simulation Acta Materialia 2001. - 49(- 17): p. - 3441.

40. Ma A, Roters F, Raabe D, A dislocation density based constitutive model for crystal plasticity FEM including geometrically necessary dislocations Acta Materialia 2006. - 54(- 8): p. - 2179.

41. Anand L, Single-crystal elasto-viscoplasticity: application to texture evolution in polycrystalline metals at large strains Computer Methods in Applied Mechanics and Engineering 2004. - 193(- 48-51): p. - 5383.

42. Kothari M, Anand L, Elasto-viscoplastic constitutive equations for polycrystalline metals: Application to tantalum Journal of the Mechanics and Physics of Solids 1998. - 46(- 1): p. - 67.

43. Li S, Van Houtte P, Kalidindi SR, A quantitative evaluation of the deformed texture predictions for aluminium alloys from crystal plasticity finite element method.. Modelling and Simulation in Materials Science and Engineering, 2004. 12: p. 845-870.

44. Bhattacharyya A, El-Danaf E, Kalidindi, SR, Doherty RD, Evolution of grain-scale microstructure during large strain simple compression of polycrystalline aluminum with quasi-columnar grains: OIM measurements and numerical simulations. International Journal of Plasticity, 2001. 17(6): p. 861-883.

45. Delannay L, Kalidindi SR, Van Houtte P, Quantitative prediction of textures in aluminium cold rolled to moderate strains. Materials Science and Engineering A, 2002. 336(1-2): p. 233-244.

46. Beaudoin, AJ, et al. Three-dimensional deformation process simulation with explicit use of polycrystal plasticity models. International Journal of Plasticity 1993. - 9(- 7): p. - 860.

47. Kalidindi SR, Bhattacharya A, Doherty RD, Detailed Analysis of Plastic Deformation in Columnar Polycrystalline Aluminum Using Orientation Image Mapping and Crystal Plasticity Models. Proceedings of the Royal Society of London: Mathematical, Physical and Engineering Sciences., 2004. 460(2047): p. 1935 - 1956

48. Wenk H-R, Van Houtte P, Texture and Anisotropy. Reports on Progress in Physics, 2004. 67: p. 1367-1428.

49. Bohlke T, Jensen, Bertram A, The evolution of Hooke's law due to texture development in FCC polycrystals. International Journal of Solids and Structures, 2001. 38(52): p. 9437-9459.

50. Li DS, Garmestani H, Schoenfeld S, Evolution of crystal orientation distribution coefficients during plastic deformation. Scripta Materialia, 2003. 49: p. 867-872.

51. Li S, Van Bael A, Van Houtte P, Teodosiu C, Finite element modeling of plastic anisotropy induced by texture and strain-path change. International Journal of Plasticity, 2003. 19(5): p. 647-674.

52. Kalidindi SR, Duvvuru HK, Spectral methods for capturing crystallographic texture evolution during large plastic strains in metals. Acta Materialia, 2005. 53 (13): p. 3613-3623.

53. Kalidindi SR, Duvvuru HK, Knezevic M, Spectral calibration of crystal plasticity models. Acta Materialia, 2006. 54(7): p. 1795-1804.

54. Bunge H, Esling C, Texture Development by Plastic Deformation. Scripta Metall, 1984. 18: p. 191-195.

55. Coulomb A, Clement P, Eulerian Simulation of Deformation Textures. Scripta Metallurgica, 1979. 13: p. 899-901.

56. Van Houtte P, Application of plastic potentials to strain rate sensitive and insensitive anisotropic materials. International Journal of Plasticity, 1994. 10(7): p. 719-748.

57. Knezevic M, PhD Thesis in progress, Drexel University, USA, 2008.

58. Knezevic M, Kalidindi SR, Fullwood D, Computationally efficient database and spectral interpolation for crystal plasticity calculations: Application to face-centered cubic polycrystals. submitted for review, 2007.

59. Hill R, A Theory of the Yielding and Plastic Flow of Anisotropic Metals. Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences, 1948. 193(1033): p. 281-297.

60. Hill R, Constitutive modelling of orthotropic plasticity in sheet metals. Journal of the Mechanics and Physics of Solids, 1990. 38(3): p. 405-417.

61. Barlat F, Crystallographic texture, anisotropic yield surfaces and forming limits of sheet metals Materials Science and Engineering 1987. - 91(-): p. - 72.

62. Barlat F, Lian K, Plastic behavior and stretchability of sheet metals. Part I: A yield function for orthotropic sheets under plane stress conditions International Journal of Plasticity 1989. - 5(- 1): p. - 66.

63. Houskamp JR, Microstructure Sensitive Design: A Tool for Exploiting Material Anisotropy in Mechanical Design. PhD Thesis, Drexel University, USA, 2005.

APPENDIX A: FORTRAN CODE TO COMPUTE THE SPECTRAL DATABASES DESCRIBED IN BUNGE_ESLING APPROACH

```
C***********************************************************************
****
C   THIS PROGRAMS COMPUTES the constants
```
$A_{l\lambda}^{\mu\nu\sigma\rho}$ and $B_l^{\mu\nu}$ in equation

$$F_l^{\mu\nu}(N\Delta\eta)=\sum_{\lambda=1}^{\infty}\sum_{\sigma=1}^{M(\lambda)}\sum_{\rho=1}^{N(\lambda)}A_{l\lambda}^{\mu\nu\sigma\rho}F_\lambda^{\sigma\rho}((N-1)\Delta\eta)+B_l^{\mu\nu}$$ described in Bunge-Esling

approach in Chapter 5

```
C
C***********************************************************************
****
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(MAXCRYS=2000,MAXDIM=10000)
      DIMENSION CCOEFF0(MAXCRYS,MAXDIM),CCOEFF1(MAXCRYS,MAXDIM)

      DIMENSION cout(maxcrys,maxdim),each_crystal(MAXDIM)

      DIMENSION AMAT(MAXDIM,MAXDIM),CNEW(MAXDIM),AINV(MAXDIM,MAXDIM),
     & INDX(MAXDIM),CONS(MAXDIM),A(MAXDIM,MAXDIM),C(MAXDIM)
C
C
      TINY = .001
C
      OPEN(10,FILE='Ccoeff192_Taylortex_2000Initial.txt',STATUS='OLD')
      OPEN(11,FILE='Ccoeff192_100pts_Taylrtex_10%.txt',STATUS='OLD')
      OPEN(15,FILE='INFO')
      OPEN(20,FILE='Cout')
      OPEN(66,FILE='Big AMAT')
      OPEN(33,FILE='C')
      OPEN(88,FILE='crystal')
100   FORMAT(A20)
      write(*,*)'Enter No. of Crystals'
      read(*,*)NCRYS
      WRITE(*,*)'ENTER NO. OF DIMENSIONS'
      READ(*,*)NDIM
      NDIM2=NDIM**2
      IF(NCRYS.GT.MAXCRYS)WRITE(*,*)'ERROR:NCRYS>MAXCRYS'
      IF(NDIM2.GT.MAXDIM)WRITE(*,*)'ERROR:NDIM2>MAXDIM'
C     Reading the X and Y vectors and writing them to CCOEFFO and
CCOEFF1 arrays respectively
      DO 102 ICRYS=1,NCRYS
            READ(10,*)(CCOEFF0(ICRYS,J),J=1,NDIM)
            WRITE(22,*)(CCOEFF0(ICRYS,J),J=1,NDIM)
            READ(11,*)(CCOEFF1(ICRYS,J),J=1,NDIM)
102         WRITE(22,*)(CCOEFF1(ICRYS,J),J=1,NDIM)
C
      DO 103 K=1,(NDIM2+NDIM)
            CNEW(K)=0.0
      DO 103 L=1,(NDIM2+NDIM)
103         AMAT(K,L)=0.0
```

```
CC    Populating the AMAT matrix with NDIM2*NDIM2 elements
      DO 400 ICRYS=1,NCRYS
      DO 400 K=1,NDIM2
            IA= (K-1)/NDIM+1
            IB= K-((K-1)/NDIM)*NDIM
            CNEW(K)=CNEW(K)+CCOEFF1(ICRYS,IA)*CCOEFF0(ICRYS,IB)
      DO 400 L=(IA-1)*NDIM+1,IA*NDIM
400         AMAT(K,L)=AMAT(K,L)+CCOEFF0(ICRYS,L-(IA-
1)*NDIM)*CCOEFF0(ICRYS,IB)
C     Populating the rest of AMAT with (NDIM2+NDIM)*(NDIM2+NDIM)
elements
      DO 8000 ICRYS=1,NCRYS
            itemp1=0
            t=1
            s=ndim
      DO 8000 K=(NDIM2+1),(NDIM2+NDIM)
            itemp1=itemp1+1
            CNEW(K)=CNEW(K)+CCOEFF1(ICRYS,itemp1)
            itemp2=0
                  IF(K .EQ. (NDIM2+1)) THEN
                        DO 4444 L=1,NDIM
                        itemp2=itemp2+1
                        AMAT(K,L)=AMAT(K,L)+CCOEFF0(ICRYS,itemp2)
4444                    AMAT(L,K)=AMAT(K,L)
                  ELSE
                        t=t+ndim
                        s=s+ndim
                        DO 5555 L=t,s
                        itemp2=itemp2+1
                        AMAT(K,L)=AMAT(K,L)+CCOEFF0(ICRYS,itemp2)
5555                    AMAT(L,K)=AMAT(K,L)
                  END IF
8000  continue
C     Diagonal elements(NDIM2+NDIM) of AMAT
      DO 4500 K=(NDIM2+1),(NDIM2+NDIM)
4500        AMAT(K,K)=NCRYS
C
      DO 421 i=1,ndim2+ndim
421         write(*,422)(amat(i,j),j=1,ndim2+ndim)
422   format(9e10.2)
cc
      DO 4221 i=1,ndim2+ndim
4221        write(66,422)(amat(i,j),j=1,ndim2+ndim)


      CALL MATINV(AMAT,AINV,INDX,CONS,MAXDIM,NDIM2+NDIM)
C
      DO 416 IDIM=1,NDIM2+NDIM
        CONS(IDIM)=0.0
      DO 416 JDIM=1,NDIM2+NDIM
416     CONS(IDIM)=CONS(IDIM)+AINV(IDIM,JDIM)*CNEW(JDIM)
C     A matrix
      DO 417 I=1,NDIM
      DO 417 J=1,NDIM
417   A(I,J)=CONS((I-1)*NDIM+J)
      DO 418 I=1,NDIM
418   WRITE(15,999)(A(I,J),J=1,NDIM)
```

```
C     Constant C matrix
      j=ndim2
      DO 4177 I=1,NDIM
            j=j+1
4177        C(I)=CONS(j)
      WRITE(33,999)(C(I),I=1,NDIM)
C
C     ******* Multiplying I/P with A and printing O/p for vector plots
*****
      IDIM=1
      JDIM=1
      DO 212 ICRYS=1,NCRYS
            cout(icrys,idim)=0.0
            DO 213 IDIM=1,NDIM
      cout(icrys,idim)=cout(icrys,idim)+C(idim)
              DO 211 JDIM=1,NDIM
211
      cout(icrys,idim)=cout(icrys,idim)+A(IDIM,JDIM)*CCOEFF0(ICRYS,JDIM
)
213       continue
212   continue
      DO 312 icrys=1,ncrys
999   format(1X,100F20.14)
312   WRITE(20,999)(cout(icrys,j),j=1,ndim)
cc
c     *********** Calculating the error *********************
      ERR=0.0
      nrm_err=0.0
      tot_dist=0.0
      tot_errdist=0.0
      avg_dist=0.0
      total_dist=0.0
c     Calculating the distance between intial point (co) and final
point (ci) and taking the average
      DO 500 ICRYS=1,NCRYS
      dist = 0.0
      DO 501 IDIM=1,NDIM
      CI = CCOEFF1(ICRYS,IDIM)
501   dist =dist+(ci-ccoeff0(icrys,idim))**2
      dist=dsqrt(dist)
500   tot_dist=tot_dist+dist
      avg_dist=(tot_dist)/ncrys
c     Calculating the distance between original final point
(ci,taylor)and calculated final point (cout ,A matrix)
c      and taking the average
      DO 5000 ICRYS=1,NCRYS
      ERRDIST=0.0
      DO 5001 IDIM=1,NDIM
      CI = CCOEFF1(ICRYS,IDIM)
5001  ERRDIST=ERRDIST+(CI-cout(icrys,idim))**2
      ERRDIST = DSQRT(ERRDIST)
      tot_errdist=tot_errdist+ERRDIST
      each_crystal(ICRYS)=(ERRDIST)/(avg_dist)
5000  WRITE(88,*)(each_crystal(ICRYS))
      avg_err=(tot_errdist)/ncrys
      nrm_err=(avg_err)/(avg_dist)
      write(*,*) 'Avg dist bet initial and final point=',avg_dist
      write(*,*) 'Avg error bet original and cal final point=',avg_err
      write(*,*)'Norm_error,(Avg_dist)/(Avg_error)=',(avg_err)/(avg_dis
t)
120   FORMAT(6F10.5)
      STOP
      END
C((((((((((((((((((((((((((((((((((((((()))))))))))))))))))))))))))))))))))))))
C  THIS SUBROUTINE MULTIPLIES MATRIX A BY MATRIX B TO GIVE MATRIX C
      SUBROUTINE MATMUL(A,IA,B,IB,C,IC,L,M,K)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(IA,IA),B(IB,IB),C(IC,IC)
      DO 10 I=1,L
      DO 10 J=1,K
       C(I,J)=0.0
       DO 20 II=1,M
  20     C(I,J)=C(I,J)+A(I,II)*B(II,J)
10     CONTINUE
      RETURN
      END
C((((((((((((((((((((((((((((((((((((((()))))))))))))))))))))))))))))))))))))))
C  THIS SUBROUTINE ADDS MATRIX A TO MATRIX B TO GIVE MATRIX C
      SUBROUTINE MATADD(A,IA,B,IB,C,IC,L,K)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(IA,IA),B(IB,IB),C(IC,IC)
      DO 10 I=1,L
      DO 10 J=1,K
10     C(I,J)=A(I,J)+B(I,J)
      RETURN
      END
C((((((((((((((((((((((((((((((((((((((()))))))))))))))))))))))))))))))))))))))
C  THIS SUBROUTINE SUBTRACTS MATRIX B FROM MATRIX A TO GIVE MATRIX C
      SUBROUTINE MATSUB(A,IA,B,IB,C,IC,L,K)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(IA,IA),B(IB,IB),C(IC,IC)
      DO 10 I=1,L
      DO 10 J=1,K
10     C(I,J)=A(I,J)-B(I,J)
      RETURN
      END
C((((((((((((((((((((((((((((((((((((((()))))))))))))))))))))))))))))))))))))))
C  THIS SUBROUTINE EQUATES MATRIX A TO MATRIX B
      SUBROUTINE MATEQL(A,IA,B,IB,L,K)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(IA,IA),B(IB,IB)
      DO 10 I=1,L
      DO 10 J=1,K
10     B(I,J)=A(I,J)
      RETURN
      END
C(((((((((((((((((((((((((((((((((((((((())))))))))))))))))))))))))))))))))))))))
C  THIS SUBROUTINE INVERTS MATRIX A.
        SUBROUTINE MATINV(A,AINV,INDX,R,NP,N)
        IMPLICIT REAL*8(A-H,O-Z)
        DIMENSION A(NP,NP),INDX(NP),R(NP),AINV(NP,NP)
        CALL LUDCMP(A,N,NP,INDX,D)
      write(*,*)'no problem'
         DO 10 I= 1,N
         CALL VEC0(R,NP,N)
         R(I) = 1.0
         CALL LUBKSB(A,N,NP,INDX,R)
         DO 10 J = 1,N
10         AINV(J,I) = R(J)
         END
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C  THIS SUBROUTINE AND THE NEXT ONE USE THE LU DECOMPOSITION METHOD
C  TO SOLVE A SYSTEM OF ALGEBRIAC EQUATIONS
      SUBROUTINE LUDCMP(A,N,NP,INDX,D)
      PARAMETER (NMAX=8500)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(NP,NP),INDX(NP),VV(NMAX)
      TINY = 1.0D-20
      D=1.
      DO 12 I=1,N
        AAMAX=0.
        DO 11 J=1,N
          IF (ABS(A(I,J)).GT.AAMAX) AAMAX=ABS(A(I,J))
11      CONTINUE
        IF (AAMAX.EQ.0.) PAUSE 'Singular matrix.'
        VV(I)=1./AAMAX
12    CONTINUE
      DO 19 J=1,N
        IF (J.GT.1) THEN
          DO 14 I=1,J-1
            SUM=A(I,J)
            IF (I.GT.1)THEN
              DO 13 K=1,I-1
                SUM=SUM-A(I,K)*A(K,J)
13            CONTINUE
              A(I,J)=SUM
            ENDIF
14        CONTINUE
        ENDIF
        AAMAX=0.
        DO 16 I=J,N
          SUM=A(I,J)
          IF (J.GT.1)THEN
            DO 15 K=1,J-1
              SUM=SUM-A(I,K)*A(K,J)
15          CONTINUE
            A(I,J)=SUM
          ENDIF
          DUM=VV(I)*ABS(SUM)
          IF (DUM.GE.AAMAX) THEN
            IMAX=I
            AAMAX=DUM
          ENDIF
16      CONTINUE
        IF (J.NE.IMAX)THEN
          DO 17 K=1,N
            DUM=A(IMAX,K)
            A(IMAX,K)=A(J,K)
            A(J,K)=DUM
17        CONTINUE
          D=-D
          VV(IMAX)=VV(J)
        ENDIF
        INDX(J)=IMAX
        IF(J.NE.N)THEN
          IF(A(J,J).EQ.0.)A(J,J)=TINY
          DUM=1./A(J,J)
          DO 18 I=J+1,N
            A(I,J)=A(I,J)*DUM
18        CONTINUE
        ENDIF
19    CONTINUE
      IF(A(N,N).EQ.0.)A(N,N)=TINY
      RETURN
      END
c&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
      SUBROUTINE LUBKSB(A,N,NP,INDX,B)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(NP,NP),INDX(NP),B(NP)
      II=0
      DO 12 I=1,N
        LL=INDX(I)
        SUM=B(LL)
        B(LL)=B(I)
        IF (II.NE.0)THEN
          DO 11 J=II,I-1
            SUM=SUM-A(I,J)*B(J)
11        CONTINUE
        ELSE IF (SUM.NE.0.) THEN
          II=I
        ENDIF
        B(I)=SUM
12    CONTINUE
      DO 14 I=N,1,-1
        SUM=B(I)
        IF(I.LT.N)THEN
          DO 13 J=I+1,N
            SUM=SUM-A(I,J)*B(J)
13        CONTINUE
        ENDIF
        B(I)=SUM/A(I,I)
14    CONTINUE
      RETURN
      END
C*********************************************************************
C    THIS SUBROUTINE DOES THE POLAR DECOMPOSITION F=[R][U]=[V][R]
C    ANY GIVEN POSITIVE DEFINITE TENSOR F
C*********************************************************************
      SUBROUTINE DECOMP(F,R,U,N)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 IC,IIC,IIIC,IU,IIU,IIIU,ID
      DIMENSION F(N,N),FT(3,3),ID(3,3),C(3,3),CC(3,3),
     .            U(N,N),UINV(3,3),R(N,N)
C
C    TRANSPOSE F MATRIX, OBTAIN [C] = [FT] [F], [CC]= [C][C], FIND
C    PRINCIPAL INVARIANTS OF MATRIX [C].
C
      M=3
```

```fortran
      CALL MAT1(ID,3,3)
      CALL MATTRANS(F,N,FT,3,M,M)
      CALL MATMUL(FT,3,F,N,C,3,M,M,M)
      CALL MATMUL(C,3,C,3,CC,3,M,M,M)
      CALL INVARIANTS(C,3,IC,IIC,IIIC)
      CALL INVEIGEN(IC,IIC,IIIC,E1,E2,E3)
C
C    EIGEN VALUES AND INVARIANTS OF U
C
      UE1=DSQRT(E1)
      UE2=DSQRT(E2)
      UE3=DSQRT(E3)
      CALL EIGENINV(UE1,UE2,UE3,IU,IIU,IIIU)
C
C    EVALUATE COMPONENTS OF U
C
      PHI1=1.0/(IIU*(IIU*(IIU+IC)+IIC)+IIIC)
      ALP1=-(IU*IIU-IIIU)
      BETA1=(IU*IIU-IIIU)*(IIU+IC)
      GAM1=(IU*IIIC+IIIU*(IIU*(IIU+IC)+IIC))
      DO 10 I=1,M
      DO 10 J=1,M
10       U(I,J)=PHI1*(ALP1*CC(I,J)+BETA1*C(I,J)+GAM1*ID(I,J))
C
C    EVALUATE COMPONENTS OF U INVERSE
C
      PHI2=1.0/(IIIU*IIIU*(IIIU+IU*IC)+IU*IU*(IU*IIIC+IIIU*IIC))
      ALP2=IU*(IU*IIU-IIIU)
      BETA2=-(IU*IIU-IIIU)*(IIIU+IU*IC)
      GAM2=IIU*IIIU*(IIIU+IU*IC)+IU*IU*(IIU*IIC+IIIC)
      DO 20 I=1,M
      DO 20 J=1,M
20     UINV(I,J)=PHI2*(ALP2*CC(I,J)+BETA2*C(I,J)+GAM2*ID(I,J))
C
C    EVALUATE [R]=[F][UINV]
C
      CALL MATMUL(F,N,UINV,3,R,N,M,M,M)
      RETURN
      END
C**************************************************************
C  THIS SUBROUTINE CALCULATES THE INVARIANTS OF A 3X3 MATRIX
C**************************************************************
      SUBROUTINE INVARIANTS(C,ICC,IC,IIC,IIIC)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 IC,IIC,IIIC
      DIMENSION C(ICC,ICC)
C
C   FOR 3 X 3 MATRICES ONLY
C
      IC=0.0
      DO 10 I=1,3
10     IC=IC+C(I,I)
      IIC=0.0
      DO 20 I=1,3
      DO 20 K=1,3
20     IIC=IIC+C(I,K)*C(K,I)
      IIC=0.5*(IC**2-IIC)
      IIIC=C(1,1)*(C(2,2)*C(3,3)-C(2,3)*C(3,2))-
     .           C(1,2)*(C(2,1)*C(3,3)-C(2,3)*C(3,1))+
     .           C(1,3)*(C(2,1)*C(3,2)-C(2,2)*C(3,1))
      RETURN
      END
C*************************************************************
C  THIS SUBROUTINE CALCULATES THE EIGENVALUES, GIVEN THE
C  INVARIANTS OF A 3X3 SYMMETRIC MATRIX WHOSE EIGEN VALUES ARE
C  ALL POSITIVE. THE METHOD USED IS SOLVING THE CUBIC
C  CHARACTERISTIC EQUATION OF THE MATRIX. PAGE 157 OF NUMERICAL
C  RECIPES.
C*************************************************************
      SUBROUTINE INVEIGEN(IC,IIC,IIIC,E1,E2,E3)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 IC,IIC,IIIC
      IF(IC.EQ.3.0D0.AND.IIC.EQ.3.0D0.AND.IIIC.EQ.1.0D0)THEN
        E1=1.0D0
        E2=1.0D0
        E3=1.0D0
        RETURN
      ENDIF
C     B=IC
C     C=-IIC
C     D=IIIC
C     CALL CUBIC(B,C,D,E1,E2,E3)
C
C  REFORMULATE THE EQUATION INTERMS OF (E-1) AS UNKNOWN
C
      P=0.0D0
      A1=-(3.0-IC)*10.0**P
      A2=-(3.0-2.0*IC+IIC)*10.0**(2.0*P)
      A3=-(1.0-IC+IIC-IIIC)*10.0**(3.0*P)
      CALL CUBIC(A1,A2,A3,E1,E2,E3)
      E1=1.0+E1/10.0**P
      E2=1.0+E2/10.0**P
      E3=1.0+E3/10.0**P
      RETURN
      END
C******************************************************************
C   THIS SUBROUTINE SOLVES A CUBIC EQUATION:
C   X^3 = A1*X^2 +A2*X + A3
C******************************************************************
      SUBROUTINE CUBIC(A1,A2,A3,X1,X2,X3)
      IMPLICIT REAL*8(A-H,O-Z)
      Q = (A1*A1)/3.0+A2
      R = (2.0*A1*A1*A1+9.0*A1*A2+27.0*A3)/27.0
      S=2.0*DSQRT(Q/3.0)
      IF(S.EQ.0.0)THEN
        X1=0.0
        X2=0.0
        X3=0.0
      ELSE
        XARG=4.0*R/(S*S*S)
        IF(ABS(XARG).GT.1.0D0)XARG=1.0D0
        THETA=(1./3.)*DACOS(XARG)
        PI=DACOS(-1.0D0)
        X1=S*DCOS(THETA)
```

```
  X2=S*DCOS(THETA-2.*PI/3.)
  X3=S*DCOS(THETA+2.*PI/3.)
ENDIF
AA=A1/3.0
X1=X1+AA
X2=X2+AA
X3=X3+AA
RETURN
END
```

APPENDIX B: C++ CODE TO COMPUTE THE TEXTURE EVOLUTION AND THE CONCOMITANT ANISOTROPIC STRESS_STRAIN RESPONSE IN A GIVEN DEFORMATION PROCESS USING THE SPECTRAL CRYSTAL PLASTICITY APPROACH.

```
/***
      The original code was from Steve Sintay (Brigham Young
      University),to calculate the Fourier coefficients for cubic
      system. 06/19/2004
      Modified by Hari Duvvuru to incorporate spectral crystal
      plasticity.
      The modified files msdi.cpp, material.cpp and material.h are
      presented in this appendix. Also, some of the C++ methods used in
      the calculation of Fourier coefficients that are not modified
      from the original project in material.cpp are not reproduced
      here.
      Other required files for the entire project to execute  are not
      provided here as they are not modified.
      Last modified 02/19/2007.
***/

***** Msdi.cpp ****************************************************

#include <stdio.h>
#include <stdlib.h>
#include <fstream>
#include <iostream>
#include <string>
#include <vector>
#include "material.h"
using namespace std;
int main()
{
double InitialAngle[2000][3];
double W21Re[2000],W21Im[2000],W31Re[2000],W31Im[2000],W32Re[2000],
       W32Im[2000];
double Wstar[9],GF[3][3],b[3];
//double DeltaT=20.00*1.414213562;
double DeltaT=10.0*1.414213562;//for shear
int i,j,nterms,nsteps,Totcrys;
cout<<"Enter no. of crystals(min=1): "<<endl;
cin>>Totcrys;
cout<<"Enter no. of strain steps: "<<endl;
cin>>nsteps;
//Create a material object
Material m;

//////////////////////////////////////////////////////////////////
/*The SetSymmetry() function will accept the following values for
SampleSym and CrystalSym. The values must
      of course be integers so the values must be #defined or "enumed"
in some other part of the program. In
```

```
      this case they are defined in the header file TexCalcGSHE_sds.h.
If a symmetry value is given that is not
      know then the default is to set the symmetry to unknown.*/
// SampleSym Only:                              SampleSym and
CrystalSym:
//
//          SSYM_TRICLINIC                      OH                cubic
//          SSYM_ORTHOTROPIC              D4H
      ditetragonal
//          SSYM_AXIAL                          D2H
      orthrohombic
//                                                    D6H
      dihexagonal
//                                                    D3D
      ditrigonal
//                                                    CIs
      triclinic
m.SetSymmetry(OH,CIs); //Cubic Triclinic
// m.SetSymmetry(OH, D2H); // Cubic Orthorhombic
//Set the rank
m.SetRank(10);
      //Initialize all the stuff that depends on symmetry and rank
      if(!m.Init(m.CrystalSym(),m.SampleSym()))
      {     cout << "Material init failed" << endl;
            return(1);
      }
      for(int z=0;z<Totcrys;z++){//Intialization
      W21Re[z]=0.0;W21Im[z]=0.0;
      W31Re[z]=0.0;W31Im[z]=0.0;
      W32Re[z]=0.0;W32Im[z]=0.0;
      }
ifstream initialOrient("Initial_1000.txt");
      if(!initialOrient.is_open()){
      cout<<"Error openeing Initial_1000.txt file"<<endl; exit(1);
            }
ofstream Wfile("Wstar-Spectral.txt");//output file for Acoeffs
ofstream Texfile("Texture.txt");//output file for Texture
ofstream Stfile("AvgStress.txt");//output file for stress
ofstream Crysfile("CrystalStress.txt");//output file for Crysstress
Stfile<<"Average Stress"<<endl;
for(int n=0;n<nsteps;n++){
      cout<<"n="<<n<<endl;
      Crysfile<<n<<endl;
for(i=0;i<Totcrys;i++){
      if(n==0){
initialOrient>>InitialAngle[i][0]>>InitialAngle[i][1]>>InitialAngle[i][
2];
         }
m.Summation(Wstar,InitialAngle[i][0],InitialAngle[i][1],InitialAngle[i]
[2],n,i);
            for(j=0;j<9;j++){
            Wstar[j]=DeltaT*Wstar[j];
            Wsfile<<Wstar[j]<<"\t";
            }
            Wsfile<<endl;;
m.AngleAxis(Wstar,GF,InitialAngle[i][0],InitialAngle[i][1],InitialAngle
[i][2]);
```

```
            m.EulerAngles(GF,b);
            if(n==nsteps-1){
            Texfile<<b[0]<<"\t"<<b[1]<<"\t"<<b[2]<<endl;
            }
InitialAngle[i][0]=b[0];InitialAngle[i][1]=b[1];InitialAngle[i][2]=b[2]
;
}
m.AvgStress11Re=m.AvgStress11Re/(100.0*Totcrys);
m.AvgStress11Im=m.AvgStress11Im/(100.0*Totcrys);
m.AvgStress22Re=m.AvgStress22Re/(100.0*Totcrys);
m.AvgStress22Im=m.AvgStress22Im/(100.0*Totcrys);
m.AvgStress33Re=m.AvgStress33Re/(100.0*Totcrys);
m.AvgStress33Im=m.AvgStress33Im/(100.0*Totcrys);
Stfile<<m.AvgStress11Re<<"\t"<<m.AvgStress22Re<<"\t"<<m.AvgStress33Re<<
endl;
 }
for(j=0;j<Totcrys;j++){
Gammafile<<m.GammaRe_t[j]<<"\t"<<m.GammaIm_t[j]<<endl;
}     initialOrient.close();
      Wfile.close();
      Stfile.close();
}

***** material.h ****************************************************
// material.h: interface for the Material class.
//
//////////////////////////////////////////////////////////////////////
#ifndef MATERIAL_HPP
#define MATERIAL_HPP

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <vector>
#include <string>
#include <iostream>
#include <fstream>


#include "point.h"
#include "TexCalcGSHE_sds.h"

class Material : public CTexCalcGSHE
{
public:

      Material(void);
      ~Material(void);

void FourIndicesFromTwo(int q, int t, int &i, int &j, int &r, int &s);
//Helper functions to setup the symmetry of the material
void SetSymmetry(int CrystalSym, int SampleSym);
int CrystalSym(){return(m_crystalSym);};
int SampleSym(){return(m_sampleSym);};
void getName(std::string &name);
void setName(std::string name);
gshe_complex Kijrs[6][6][10], Ksijrs[6][6][10];
gshe_complex *****mStiffCoeff, *****mCompCoeff;
double mS[6][6], mC[6][6], mC11, mC12, mC44, mS11, mS12, mS44, Cg;
double AvgStress11Re,AvgStress11Im,AvgStress22Re,AvgStress22Im,
       AvgStress33Re,AvgStress33Im;
double W21AlmnRe[1000],W21AlmnIm[1000],W31AlmnRe[1000],W31AlmnIm[1000],
       W32AlmnRe[1000],W32AlmnIm[1000];
double S11Blmnk1Re[29][80],S22Blmnk1Re[29][80],S33Blmnk1Re[29][80];
double S11Blmnk1Im[29][80],S22Blmnk1Im[29][80],S33Blmnk1Im[29][80];
double W21Blmnk1Re[29][80],W31Blmnk1Re[29][80],W32Blmnk1Re[29][80];
double W21Blmnk1Im[29][80],W31Blmnk1Im[29][80],W32Blmnk1Im[29][80],
       GaBlmnk1Re[29][80],GaBlmnk1Im[29][80];
double W21Re,W21Im,W31Re,W31Im,W32Re,W32Im;
double GlmnRe[1000],GlmnIm[1000],GammaRe,GammaIm,SgRe,SgIm;
double Slmn11Re[1000],Slmn11Im[1000],Sigma11Re,Sigma11Im;
double Slmn33Re[1000],Slmn33Im[1000],Sigma33Re,Sigma33Im;
double Slmn22Re[1000],Slmn22Im[1000],Sigma22Re,Sigma22Im;
double GammaRe_t[1000],GammaIm_t[1000],SRe_t[1000],SIm_t[1000],k[29],
       theta[10];
double So,ho,Ss,a;double rs;
double xRe,xIm;
double ts1,ts2;
double CrysStress11Re,CrysStress11Im,CrysStress33Re,
       CrysStress33Im,CrysStress22Re,CrysStress22Im;
double W21TmpRe,W21TmpIm, W31TmpRe,W31TmpIm, W32TmpRe,W32TmpIm;
double S11TmpRe,S11TmpIm, S22TmpRe,S22TmpIm, S33TmpRe,S33TmpIm,
GaTmpRe,GaTmpIm;

void Summation(double A[9],double ,double ,double, int,int );
void Material::AngleAxis(double A[9],double
GF[3][3],double,double,double);
void Material::RotaMat(double ,double B[3],double R[3][3]);
void Material::EulerAngles(double GF[3][3],double b[]);

protected:

      //Helper functions and variables for output of Triclinic Terms
      void ReduceTricTerms();
      void ExpandTricTerms();
      void AllocateTricTerms();
      void FreeTricTerms();
      bool TricTermsAllocReady;
      gshe_complex ***mTricTerms;
      //Helper functions and variable for calculation of Stiff Coeff
      bool StiffAllocReady, StiffCoeffReady;
      //Helper functions and variable for calculation of Comp Coeff
      void AllocateCompCoeff(void);
      void FreeCompCoeff(void);
      bool CompAllocReady, CompCoeffReady;
//    double x[6], c[6];
      std::string mName;
      double g1[3][3];
      int m_sampleSym, m_crystalSym;
      int mHullPointCount, mHullDim;
private:
};
```

```
#endif

***** material.cpp ****************************************************
// material.cpp: implementation of the Material class.
//////////////////////////////////////////////////////////////////////
#include <stdio.h>
#include <stdlib.h>
#include <fstream.h>
#include <iostream.h>
#include "material.h"
/*
Class constructor to initiallize certain values that will be
needed by other elements of the class.
*/
Material::Material(void)
{
      TricTermsAllocReady = false;
      AvgStress11Re=0.0,AvgStress11Im=0.0;
      AvgStress33Re=0.0,AvgStress33Im=0.0;
      AvgStress22Re=0.0,AvgStress22Im=0.0;
      So=16;
      ho=180;
      Ss=148;
      a=2.25;
      rs=0.01;
      //k=pow(0.5,rs);
ifstream updatefile1("slipresupdate.txt");
    for(int i=0;i<1000;i++){
      GammaRe_t[i]=0.0,GammaIm_t[i]=0.0;
      updatefile1>>ts1 >>ts2;
      //SRe_t[i]=So,SIm_t[i]=0.0;
      SRe_t[i]=ts1,SIm_t[i]=ts2;
      }
ifstream Afile1("W21AlmnCoeffs.txt");
      if(!Afile1.is_open()){
      cout<<"Error openeing W21AlmnCoeffs.txt file"<<endl; exit(1);
            }
ifstream Afile2("W31AlmnCoeffs.txt");
      if(!Afile2.is_open()){
      cout<<"Error openeing W31AlmnCoeffs.txt file"<<endl; exit(1);
            }
ifstream Afile3("W32AlmnCoeffs.txt");
      if(!Afile3.is_open()){
      cout<<"Error openeing W32AlmnCoeffs.txt file"<<endl; exit(1);
            }
ifstream Afile4("Slmn11Coeffs.txt");
      if(!Afile4.is_open()){
      cout<<"Error openeing SlmnCoeffs.txt file"<<endl; exit(1);
            }
ifstream Afile4a("Slmn33Coeffs.txt");
      if(!Afile4a.is_open()){
      cout<<"Error openeing Slmn33Coeffs.txt file"<<endl; exit(1);
            }
ifstream Afile4b("Slmn22Coeffs.txt");
      if(!Afile4b.is_open()){
      cout<<"Error openeing Slmn22Coeffs.txt file"<<endl; exit(1);
```

```
            }
ifstream Afile5("GlmnCoeffs.txt");
      if(!Afile5.is_open()){
      cout<<"Error openeing GlmnCoeffs.txt file"<<endl; exit(1);
            }
ifstream ip("lmnfile.txt");
      if(!ip.is_open()){
      cout<<"Error openeing lmnfile.txt file"<<endl; exit(1);
            }
      //for(int i=0;i<80;i++){
      for(int i=0;i<29;i++){// number of coefficients k
            for(int j=0;j<80;j++){
      Afile1>> k[i] >> W21Blmnk1Re[i][j] >> W21Blmnk1Im[i][j];
      Afile2>> k[i] >> W31Blmnk1Re[i][j] >> W31Blmnk1Im[i][j];
      Afile3>> k[i] >> W32Blmnk1Re[i][j] >> W32Blmnk1Im[i][j];
      Afile4>> k[i] >> S11Blmnk1Re[i][j] >> S11Blmnk1Im[i][j];
      Afile4b>> k[i] >> S22Blmnk1Re[i][j] >> S22Blmnk1Im[i][j];
      Afile4a>> k[i] >> S33Blmnk1Re[i][j] >> S33Blmnk1Im[i][j];
      Afile5>> k[i] >> GaBlmnk1Re[i][j]  >> GaBlmnk1Im[i][j];
            }
      }
      Afile1.close();Afile2.close();Afile3.close();
      Afile4.close();Afile4a.close();Afile5.close();Afile4b.close();
}
/*
Class destructor used to clean up any memory leaks and other
stuff when the class is dropped from scope.
*/
Material::~Material()
{
      DataPoints.clear();
      FreeTricTerms();
}

ifstream ip ("lmnfile.txt");
int l,mu,nu;

void Material::getName(std::string &x)
{
    x = mName;
}
/*
Set the name of the material
*/
void Material::setName(std::string x)
{
    mName = x;
}

void Material::Summation(double Wstar[],double phi1,double PHI,double
phi2,int flag,int ncrys)
{
int counter=0;
//begin of the loops for l,m,n:
//cout<<"phi1= "<<phi1<<"PHI= "<<PHI<<"phi2= "<<phi2<<endl;
GammaRe=0.0,GammaIm=0.0,Sigma11Re=0.0,Sigma11Im=0.0,Sigma33Re=0.0,Sigma
33Im=0.0;
```

```
Sigma22Re=0.0,Sigma22Im=0.0;
W21Re=0.0,W21Im=0.0,W31Re=0.0,W31Im=0.0,W32Re=0.0,W32Im=0.0;
CrysStress11Re=0.0,CrysStress11Im=0.0;
CrysStress22Re=0.0,CrysStress22Im=0.0;
CrysStress33Re=0.0,CrysStress33Im=0.0;
ofstream op ("e.txt");
       for (int l=0; l<=m_Rank; ++l){
                   for (int mu=m_Mstart; mu<=m_MLinEq[l]; ++mu){
                         for(int nu=m_Nstart; nu<=m_NLinEq[l]; ++nu){

int n=0; theta[n]=0.0;
W21TmpRe=0.0,W21TmpIm=0.0, W31TmpRe=0.0,W31TmpIm=0.0,
W32TmpRe=0.0,W32TmpIm=0.0;
S11TmpRe=0.0,S11TmpIm=0.0, S22TmpRe=0.0,S22TmpIm=0.0,
S33TmpRe=0.0,S33TmpIm=0.0, GaTmpRe=0.0, GaTmpIm=0.0;
for(int iii=0;iii<29;iii++){ // number of coefficients k
W21TmpRe=W21TmpRe+(W21Blmnk1Re[iii][counter]*cos(k[iii]*theta[n])-
W21Blmnk1Im[iii][counter]*sin(k[iii]*theta[n]));
W21TmpIm=W21TmpIm+(W21Blmnk1Re[iii][counter]*sin(k[iii]*theta[n])+W21Bl
mnk1Im[iii][counter]*cos(k[iii]*theta[n]));

W31TmpRe=W31TmpRe+(W31Blmnk1Re[iii][counter]*cos(k[iii]*theta[n])-
W31Blmnk1Im[iii][counter]*sin(k[iii]*theta[n]));
W31TmpIm=W31TmpIm+(W31Blmnk1Re[iii][counter]*sin(k[iii]*theta[n])+W31Bl
mnk1Im[iii][counter]*cos(k[iii]*theta[n]));

W32TmpRe=W32TmpRe+(W32Blmnk1Re[iii][counter]*cos(k[iii]*theta[n])-
W32Blmnk1Im[iii][counter]*sin(k[iii]*theta[n]));
W32TmpIm=W32TmpIm+(W32Blmnk1Re[iii][counter]*sin(k[iii]*theta[n])+W32Bl
mnk1Im[iii][counter]*cos(k[iii]*theta[n]));

S11TmpRe=S11TmpRe+(S11Blmnk1Re[iii][counter]*cos(k[iii]*theta[n])-
S11Blmnk1Im[iii][counter]*sin(k[iii]*theta[n]));
S11TmpIm=S11TmpIm+(S11Blmnk1Re[iii][counter]*sin(k[iii]*theta[n])+S11Bl
mnk1Im[iii][counter]*cos(k[iii]*theta[n]));

S22TmpRe=S22TmpRe+(S22Blmnk1Re[iii][counter]*cos(k[iii]*theta[n])-
S22Blmnk1Im[iii][counter]*sin(k[iii]*theta[n]));
S22TmpIm=S22TmpIm+(S22Blmnk1Re[iii][counter]*sin(k[iii]*theta[n])+S22Bl
mnk1Im[iii][counter]*cos(k[iii]*theta[n]));

S33TmpRe=S33TmpRe+(S33Blmnk1Re[iii][counter]*cos(k[iii]*theta[n])-
S33Blmnk1Im[iii][counter]*sin(k[iii]*theta[n]));
S33TmpIm=S33TmpIm+(S33Blmnk1Re[iii][counter]*sin(k[iii]*theta[n])+S33Bl
mnk1Im[iii][counter]*cos(k[iii]*theta[n]));

GaTmpRe=GaTmpRe+(GaBlmnk1Re[iii][counter]*cos(k[iii]*theta[n])-
GaBlmnk1Im[iii][counter]*sin(k[iii]*theta[n]));
GaTmpIm=GaTmpIm+(GaBlmnk1Re[iii][counter]*sin(k[iii]*theta[n])+GaBlmnk1
Im[iii][counter]*cos(k[iii]*theta[n]));
}
                  W21AlmnRe[counter]=W21TmpRe;
                  W21AlmnIm[counter]=W21TmpIm;
                  W31AlmnRe[counter]=W31TmpRe;
                  W31AlmnIm[counter]=W31TmpIm;
                  W32AlmnRe[counter]=W32TmpRe;
                  W32AlmnIm[counter]=W32TmpIm;
```

```
                    Slmn11Re[counter]=S11TmpRe;
                    Slmn11Im[counter]=S11TmpIm;
                    Slmn22Re[counter]=S22TmpRe;
                    Slmn22Im[counter]=S22TmpIm;
                    Slmn33Re[counter]=S33TmpRe;
                    Slmn33Im[counter]=S33TmpIm;
                    GlmnRe[counter]=GaTmpRe;
                    GlmnIm[counter]=GaTmpIm;

        double TReal=(symmetricT(l,mu,nu,phi1,PHI,phi2,true)).re;
        double TImag=(symmetricT(l,mu,nu,phi1,PHI,phi2,true)).im;
        //Tfile<<TReal<<"\t"<<TImag<<"\t";

        W21Re=W21Re+W21AlmnRe[counter]*TReal-W21AlmnIm[counter]*TImag;
        W21Im=W21Im+W21AlmnRe[counter]*TImag+W21AlmnIm[counter]*TReal;

        W31Re=W31Re+W31AlmnRe[counter]*TReal-W31AlmnIm[counter]*TImag;
        W31Im=W31Im+W31AlmnRe[counter]*TImag+W31AlmnIm[counter]*TReal;

        W32Re=W32Re+W32AlmnRe[counter]*TReal-W32AlmnIm[counter]*TImag;
        W32Im=W32Im+W32AlmnRe[counter]*TImag+W32AlmnIm[counter]*TReal;

        Sigma11Re=Sigma11Re+Slmn11Re[counter]*TReal-
        Slmn11Im[counter]*TImag;
        Sigma11Im=Sigma11Im+Slmn11Re[counter]*TImag+Slmn11Im[counter]*TRe
        al;

        Sigma33Re=Sigma33Re+Slmn33Re[counter]*TReal-
        Slmn33Im[counter]*TImag;
        Sigma33Im=Sigma33Im+Slmn33Re[counter]*TImag+Slmn33Im[counter]*TRe
        al;

        Sigma22Re=Sigma22Re+Slmn22Re[counter]*TReal-
        Slmn22Im[counter]*TImag;
        Sigma22Im=Sigma22Im+Slmn22Re[counter]*TImag+Slmn22Im[counter]*TRe
        al;

        GammaRe=GammaRe+GlmnRe[counter]*TReal-GlmnIm[counter]*TImag;
        GammaIm=GammaIm+GlmnRe[counter]*TImag+GlmnIm[counter]*TReal;

        ++counter;
                    }
              }
  }//end of the loop for l
        Wstar[0]=0.0, Wstar[1]=-W21Re, Wstar[2]=-W31Re;
        Wstar[3]=W21Re, Wstar[4]=0.0, Wstar[5]=-W32Re;
        Wstar[6]=W31Re, Wstar[7]=W32Re, Wstar[8]=0.0;
        xRe=1-(SRe_t[ncrys]/Ss);
        xIm=1-(SIm_t[ncrys]/Ss);
        //  PSC
        SgRe=SRe_t[ncrys]+ho*pow(xRe,a)*GammaRe_t[ncrys]*20.0*1.41421356;
        SgIm=SIm_t[ncrys]+ho*pow(xIm,a)*GammaIm_t[ncrys]*20.0*1.41421356;
        SRe_t[ncrys]=SgRe;
        // these four shd be executed after SgRe and  SgIm.Updates
        //cout<<"in material SRe="<<SRe_t[ncrys]<<endl;
        SIm_t[ncrys]=SgIm;
        GammaRe_t[ncrys]=GammaRe;
```

```
      GammaIm_t[ncrys]=GammaIm;
      CrysStress11Re=SgRe*Sigma11Re-SgIm*Sigma11Im;
      CrysStress11Im=SgIm*Sigma11Re+SgRe*Sigma11Im;
      AvgStress11Re=AvgStress11Re+CrysStress11Re;
      AvgStress11Im=AvgStress11Im+CrysStress11Im;
      CrysStress22Re=SgRe*Sigma22Re-SgIm*Sigma22Im;
      CrysStress22Im=SgIm*Sigma22Re+SgRe*Sigma22Im;
      AvgStress22Re=AvgStress22Re+CrysStress22Re;
      AvgStress22Im=AvgStress22Im+CrysStress22Im;
      CrysStress33Re=SgRe*Sigma33Re-SgIm*Sigma33Im;
      CrysStress33Im=SgIm*Sigma33Re+SgRe*Sigma33Im;
      AvgStress33Re=AvgStress33Re+CrysStress33Re;
      AvgStress33Im=AvgStress33Im+CrysStress33Im;
}//end of function

void Material::AngleAxis(double Wstar[],double GF[3][3],double
phi1,double PHI,double phi2)
{           double Q[3][3],axis[3],R[3][3];

         Q[0][0]=cos(phi1)*cos(phi2)-sin(phi1)*cos(PHI)*sin(phi2);
         Q[1][0]=sin(phi1)*cos(phi2)+cos(phi1)*cos(PHI)*sin(phi2);
         Q[2][0]=sin(PHI)*sin(phi2);
         Q[0][1]=-cos(phi1)*sin(phi2)-sin(phi1)*cos(PHI)*cos(phi2);
         Q[1][1]=-sin(phi1)*sin(phi2)+cos(phi1)*cos(PHI)*cos(phi2);
         Q[2][1]=sin(PHI)*cos(phi2);
         Q[0][2]=sin(phi1)*sin(PHI);
         Q[1][2]=-cos(phi1)*sin(PHI);
         Q[2][2]=cos(PHI);
double ang=sqrt(Wstar[1]*Wstar[1]+Wstar[2]*Wstar[2]+Wstar[5]*Wstar[5]);
             if(ang==0.0){
                  axis[0]=1.0;axis[1]=0.0;axis[2]=0.0;
             }
             else{
              axis[0]=Wstar[5]/ang;
          axis[1]=-Wstar[2]/ang;
              axis[2]=Wstar[1]/ang;
             }
             RotaMat(ang,axis,R);
             for(int i=0;i<3;i++){
                   for(int j=0;j<3;j++){
                         GF[i][j]=0.0;
                         for(int k=0;k<3;k++){
                         GF[i][j]=GF[i][j]+R[i][k]*Q[k][j];
                         }
                  }
             }
}//end of AngleAxis function

void Material::RotaMat(double ang,double axis[],double R[3][3]){
      R[0][0]=(1-axis[0]*axis[0])*cos(ang)+axis[0]*axis[0];
      R[0][1]=axis[0]*axis[1]*(1-cos(ang))-axis[2]*sin(ang);
      R[0][2]=axis[0]*axis[2]*(1-cos(ang))+axis[1]*sin(ang);
      R[1][0]=axis[0]*axis[1]*(1-cos(ang))+axis[2]*sin(ang);
      R[1][1]=(1-axis[1]*axis[1])*cos(ang)+axis[1]*axis[1];
      R[1][2]=axis[1]*axis[2]*(1-cos(ang))-axis[0]*sin(ang);
      R[2][0]=axis[1]*axis[2]*(1-cos(ang))-axis[1]*sin(ang);
      R[2][1]=axis[1]*axis[2]*(1-cos(ang))+axis[0]*sin(ang);
```

```
    R[2][2]=(1-axis[2]*axis[2])*cos(ang)+axis[2]*axis[2];
}//end of RotaMat function

void Material::EulerAngles(double GF[3][3],double b[]){
      if(fabs(GF[2][2])>1){
      cout<<"Error in gn[2][2]"<<endl;
      }
      b[1]=acos(GF[2][2]);
      if(GF[2][2]==1.0 || GF[2][2]==-1.0){
      b[2]=0;
      b[0] = acos(GF[0][0]);
      if (GF[1][0]<0.0) {b[0] = 2*PI-b[0];}
      }
      else{
            b[2]=atan2((GF[2][0]),(GF[2][1]));
            b[0]=atan2((GF[0][2]),(-GF[1][2]));
            }
      double TOL=0.001;
/*angles 0 and 2PI are same for Phi1 and Phi2 in the Euler space
0<=Phi1<2PI,0<=Phi<=PI,0<=Phi2<2PI.
To avoid numerical approximations angles are checked with in some
tolerance*/
      if (fabs(b[0]-0.0)<TOL)
            b[0]=0.0;
      else
      if (fabs(b[0]-2.0*PI)<TOL){b[0]=0.0;}
      if(fabs(b[2]-0.0)<TOL)
            b[2]=0.0;
      else
      if(fabs(b[2]-2.0*PI)<TOL){b[2]=0.0;}
      //force angles to be positive
      while(b[0]<0){b[0]=b[0]+(2*PI);}
      while(b[1]<0){b[1]=b[1]+(2*PI);}
      while(b[2]<0){b[2]=b[2]+(2*PI);}
}//end of EulerAngles function


void Material::SetSymmetry(int CrystalSym, int SampleSym)
{
    m_crystalSym = CrystalSym;
    m_sampleSym = SampleSym;
}
```

APPENDIX C: C++ CODE THAT COUPLES FORTRAN SUBROUTINES IN FORGE3 AND SPECTRAL CRYSTAL PLASTICITY WRITTEN IN MATLAB.

```
// TestInterface.h: interface for the TestInterface class.
//
//////////////////////////////////////////////////////////////////////
#include <iostream>



#if
!defined(AFX_TESTINTERFACE_H__48425A71_320E_4AC4_A95F_BA85B63FEE11__INC
LUDED_)
#define
AFX_TESTINTERFACE_H__48425A71_320E_4AC4_A95F_BA85B63FEE11__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000


extern "C" int _cdecl FFT_SCP(long double* LMatrix, long double*
additionalArguments);

#endif //
!defined(AFX_TESTINTERFACE_H__48425A71_320E_4AC4_A95F_BA85B63FEE11__INC
LUDED_)



// TestInterface.cpp: implementation of the TestInterface class.
//
//////////////////////////////////////////////////////////////////////
#include "TestInterface.h"

#include <matrix.h>
#include <engine.h>
#include <sstream>
#include <fstream>
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <iomanip>

static Engine* globalEngine = (Engine*)NULL;
const char* SCP_FFT_PATH = "C:\\Temp\\SCP_FFT_Forge_Test\\";
const char* SCP_FFT_COMMANDLOG =
"C:\\Temp\\SCP_FFT_Forge_Test\\commandLog.txt";

extern "C" int _cdecl SCP_FFT(long double* LMatrix, long double*
additionalArguments) {
	// Check for instance of the MATLAB Engine...
	if(globalEngine == (Engine*)NULL) {
		globalEngine = engOpen("-nojvm -nodesktop");
		if(globalEngine == (Engine*)NULL) {
			std::cerr << "Unable to start MATLAB Engine" <<
std::endl;
			return 0;
		}
	}

	// Setup LMatrix
	mxArray* mxLMatrix = mxCreateDoubleMatrix(3,3, (mxComplexity)0);
	double* dLmatrix = mxGetPr(mxLMatrix);
	if(mxLMatrix == (mxArray*)NULL) {
		std::cerr << "Unable to Create L Matrix" << std::endl;
		return 0;
	}
	for(int i = 0; i < 3; ++i) {
		for(int j = 0; j < 3; ++j) {
			dLmatrix[i*3+j] = LMatrix[i*3+j];
		}
	}

	// Setup TimeSteps
	mxArray* mxTimeStep = mxCreateDoubleMatrix(1,1, (mxComplexity)0);
	double* dTimeStep = mxGetPr(mxTimeStep);
	if(mxTimeStep == (mxArray*)NULL) {
		std::cerr << "Unable to Create TimeStep Matrix" <<
std::endl;
		return 0;
	}
	dTimeStep[0] = additionalArguments[0];

	// Setup Print Y/N
	mxArray* mxFilePrint = mxCreateDoubleMatrix(1,1,
(mxComplexity)0);
	double* dFilePrint = mxGetPr(mxFilePrint);
	if(mxFilePrint== (mxArray*)NULL) {
		std::cerr << "Unable to Create FilePrint Matrix" <<
std::endl;
		return 0;
	}
	dFilePrint[0] = additionalArguments[1];

	// Setup Print Y/N
	mxArray* mxElementTimeStep = mxCreateDoubleMatrix(1,1,
(mxComplexity)0);
	double* dElementTimeStep = mxGetPr(mxElementTimeStep);
	if(mxTimeStep == (mxArray*)NULL) {
		std::cerr << "Unable to Create FilePrint Matrix" <<
std::endl;
		return 0;
	}
	dElementTimeStep[0] = additionalArguments[3];

	// Setup Print Y/N
	mxArray* mxElementNumber = mxCreateDoubleMatrix(1,1,
(mxComplexity)0);
	double* dElementNumber= mxGetPr(mxElementNumber);
	if(mxElementNumber == (mxArray*)NULL) {
		std::cerr << "Unable to Create FilePrint Matrix" <<
std::endl;
		return 0;
	}
	dElementNumber[0] = additionalArguments[2];

	// Copy Arguments to workspace
	if(engPutVariable(globalEngine, "LMatrix", mxLMatrix)!=0) {
		std::cerr << "Unable to Copy LMatrix to workspace" <<
std::endl;
	}
	if(engPutVariable(globalEngine, "TimeStep", mxTimeStep)!=0) {
		std::cerr << "Unable to Copy TimeStep to workspace" <<
std::endl;
	}
	if(engPutVariable(globalEngine, "FilePrint", mxFilePrint)!=0) {
		std::cerr << "Unable to Copy FilePrint to workspace" <<
std::endl;
	}
	if(engPutVariable(globalEngine, "ElementNumber",
mxElementNumber)!=0) {
		std::cerr << "Unable to Copy ElementNumberto workspace" <<
std::endl;
	}
	if(engPutVariable(globalEngine, "ElementTimeStep",
mxElementTimeStep)!=0) {
		std::cerr << "Unable to Copy ElementNumberto workspace" <<
std::endl;
	}

	std::ostringstream matlabCommand;
	matlabCommand << "cd " << SCP_FFT_PATH << ";" <<
"FFT_SCP_final_theta(LMatrix,TimeStep,FilePrint,ElementNumber,ElementTi
meStep);";
#if 0
	// Build MATLAB command...
	FILE* commandLog = fopen(SCP_FFT_COMMANDLOG, "a+");
	if(commandLog != 0) {
		fprintf(commandLog, "%s\n", matlabCommand.str().c_str());
		fclose(commandLog);
	}
	return 1;
#endif

	// Evaluate with MATLAB...
	if(engEvalString(globalEngine, matlabCommand.str().c_str())!=0) {
		std::cerr << "Unable to evaluate MATLAB command:" <<
matlabCommand.str() << std::endl;
	}

	return 1;
}
```

APPENDIX D: FORGE3 FINITE ELEMENT INPUT FILE

Forge3 finite element input file to simulate simplecompression using forge3 loiv_meca subroutine.

```
! File Type:            FORGE3 V7.0 Data File
! Creator:        GLPre Version 2, 3, 0, 27-Release
! Author:
! Creation Date:  2007-05-03 09:54:24
! GLPre active language:English
! System language:      English (United States)
! Data File Name: squaredie.ref
! Data File Location:   C:\Forge3-
V6.3B\GLpre\Computations\NewProject.tsv\SquareDie\
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!

!================================= OBJECTS Block
.OBJETS
      ProjectName = NewProject
      SimulationName = SquareDie

      Fout = squaredie.out
      Fres = results\squaredie.res
      Faux = results\squaredie.vtf
      NBSD = 1
      objet 1, NAME=Billet
      objet 1, FMAY=billet.may
      objet 1, NomGen=results\billet_
      objet 1, rheol=1
      outil 1, NAME=LowerDie
      outil 2, NAME=UpperDie
.FIN OBJETS
!=================================

!================================= APPROXIMATION Block
.APPROXIMATION
      Periode_Meca = 1
.FIN APPROXIMATION
!=================================

!================================= UNITS Block
.UNITES
      MM-MPA-MM.KG.S
.FIN UNITES
!=================================

!================================= RHEOLOGY Block
.RHEOLOGIE
```

```
!!!!!!!!!!!!!!!!!!!!!
MATERIAU 1  ! (object Billet)
!!!!!!!!!!!!!!!!!!!!!
      EVP
      Thermoecroui: Hansel Spittel Nb1,
      ! Material name: AlMg1Si0, 6Cr
      ! Material type: Al-alloys
      ! Material subtype: Al-Mg-Si
      ! Properties type: cold forming
      ! Units: MPa, degC
      ! Validity domain:
      ! Temperature: 20 - 250
      ! Strain: 0.04 - 3
      ! Strain rate: 0 - 500
      A1=260.49451,
      m1=-0.00168,
      m2=0.16992,
      m3=0.0184,
      m4=0.00073,
      m5=0,
      m6=0,
      m7=0,
      m8=0,
      m9=0,
      eps_ss=0

      !Elasticity coefficients
      Youngmodulus = 7.300000e+04
      Poissoncoeff = 0.300000

      !Thermal coefficients
      mvolumique = 2.800000e-06 !Density
      cmassique = 1.230000e+09 !Specific Heat
      conductmat = 2.500000e+05 !Conductivity
      epsilon = 5.000000e-02 !Emissivity

!--------------------------------
OUTIL1            !LowerDie

      !Friction between deformable object and rigid die
      bilateral collant       ! Friction Law

      Temp = 20.000000
FIN OUTIL
!--------------------------------

!--------------------------------
OUTIL2            !UpperDie

      !Friction between deformable object and rigid die
      bilateral collant       ! Friction Law

      Temp = 20.000000
FIN OUTIL
!--------------------------------

      !Thermal Exchange between deformable object and air
      AlphaText = 1.000000e+001          ! Global Transfert Coeff.
      TempExt = 20.000000                ! Ambient Temperature

      ! Initial temperature has been set in mesh file: already exists
in mesh file
!!!!!!!!!!!!!!!!!!!!!
FIN MATERIAU
!!!!!!!!!!!!!!!!!!!!!

 ! *** User Variable : Sigma1

   LOIV MECA
      Sigma1
      Par STRESSTENSOR(6) = EXIST
      Var SIG1 = 0.
   FIN LOI

   ! *** User Variable Law: LatAndCN
   LOIV UTIL
      LatAndCN
      Par SIG1 = EXIST
      Par STRAIN_RATE = EXIST
      Par EQ_STRESS = EXIST
      Eta LATANDCN = 0.
   FIN LOI

   ! *** User Variable Law: Sig1_direction
   LOIV MECA
      Sig1_direction
      Par STRESSTENSOR(6) = EXIST
      Var SIG1_VECTOR(3) = 0, 0, 0
   FIN LOI
 ! *** User Variable Law: VELOCITY_GRADIENT
   LOIV MECA
      VEL_GRAD
      VAR VELGRAD1(3) = 0,0,0
      VAR VELGRAD2(3) = 0,0,0
      VAR VELGRAD3(3) = 0,0,0
  FIN LOI

!Stock=VELGRAD1,VELGRAD2,VELGRAD3

.FIN RHEOLOGIE
!================================

!================================ TOLERCONV Block
.TOLERCONV
.FIN TOLERCONV
!================================

!================================ INCREMENT Block
.INCREMENT
      Deformation= 1.000000e-002
.FIN INCREMENT
!================================

!================================ EXECUTION Block
```

```
.EXECUTION
      Inertia
      dhSto = 1.666667e-002
      Fine Sto = 1.000000e+000
      Calcul Outillage
      Folds_Detection
.FIN EXECUTION
!=================================

!================================= THERMAL Block
.THERMIQUE
.FIN THERMIQUE
!=================================

!================================= MESH BOXES Block
.BOITE

.FIN BOITE
!=================================

!================================= SENSORS Block
.CAPTEURS

.FIN CAPTEURS
!=================================

!================================= BOUNDARY CONDITIONS Block
.CONDLIM

.FIN CONDLIM
!=================================

!================================= REMESHING Block
.MAUTO

OBJET1
      periode = 50
      lbase = 0.155479
FIN OBJET

.FIN MAUTO
!=================================

!================================= KINEMATICS Block
.CINEMAT_OUT
   Outil2         ! UpperDie
      maitre
      Axe = 3
   Fin Outil
.FIN CINEMAT_OUT

.PILOT
NbPass= 1
   Pass1
   Fin Pass
.FIN PILOT
!=================================
```

APPENDIX E: LOIV_MECA FORGE3 USERSUBROUITNE.

```
**********************************************************************
C
The   original   authors   of   the   subrouitne   are   forge3   developers
(transvalor s. A.)
The following code is added to the original LOIV_MECA subroutine code.
The entire subroutine code is not presented and only the code that is
added to the original code is given here. The input file presented in
appendix D invokes this subroutine to compute the velocity gradient
tensor at each integration point for each element and pass the same to
spectral crystal plasticity code (MATLAB code) to compute texture
evolution during simple compression deformation.
C


      elseif (nom.eq.'VEL_GRAD') then
C**********************************************************************
C Determination of  the velocity gradient tensor
C
C    LOIV MECA
C      VELOCITY_GRADIENT
C      VAR VELGRAD(9) = 0,0,0,0,0,0,0,0,0
C    FIN LOI
C**********************************************************************

      if ((nbpar.ne.0).or.(nbvar.ne.3)) goto 99
C     write(*,*) nbpar

      gs_var(1) = gradv(1,1)  ! GRAD (V1)
      gs_var(2) = gradv(1,2)
      gs_var(3) = gradv(1,3)
      gs_var(4) = gradv(2,1)  ! GRAD (V2)
      gs_var(5) = gradv(2,2)
      gs_var(6) = gradv(2,3)
      gs_var(7) = gradv(3,1)  ! GRAD (V3)
      gs_var(8) = gradv(3,2)
      gs_var(9) = gradv(3,3)
C     numIterations = numIterations + 1
      LTrace = (gradv(1,1)+gradv(2,2)+gradv(3,3))

C     Setup L Matrix to pass to MATLAB...
      nativeLMatrix(1,1) = gradv(1,1)-(1.0/3.0)*(LTrace)
      nativeLMatrix(1,2) = gradv(1,2)
      nativeLMatrix(1,3) = gradv(1,3)
      nativeLMatrix(2,1) = gradv(2,1)
      nativeLMatrix(2,2) = gradv(2,2)-(1.0/3.0)*(LTrace)
      nativeLMatrix(2,3) = gradv(2,3)
      nativeLMatrix(3,1) = gradv(3,1)
      nativeLMatrix(3,2) = gradv(3,2)
      nativeLMatrix(3,3) = gradv(3,3)-(1.0/3.0)*(LTrace)
```

```
C     Create L Matrix in Matlab...

      additionalArguments(1) = tps
      additionalArguments(2) = 0.0 ! Non-zero value for updated texture
file O/P
      additionalArguments(3) = elt
      additionalArguments(4) = incr

      if(tempelt.ne.elt) then
            mxLMatrix = SCP_FFT(nativeLMatrix, additionalArguments)
      endif

      tempelt=elt
```

VITA

NAME: Hari Kishore Duvvuru

EDUCATION:

- Ph.D. Department of Materials Science and Engineering, Drexel University, Philadelphia, PA, USA, 2007.
- M.S. Department of Mechanical Engineering, South Dakota School of Mines and Technology, Rapid City, SD, USA.
- B.E. Department of Mechanical Engineering, Osmania University, Hyderabad, India.

HONORS AND ACHIEVEMENTS

- The Arthur E. Focke LeaderShape Award, awarded by ASM Materials Education Foundation, 2005.
- TMS Travel scholarship, 2004 sponsored by TMS technical divisions.
- Engineering Intern, State of South Dakota Board of Technical Professions, May 2003.
- Ivanhoe Excellence Fellowship for outstanding performance as a graduate student (2002-2003), South Dakota School of Mines and Technology.

PROFESSIONAL AFFILIATIONS

The Minerals, Metals and Materials Society (TMS), ASM International,
Material Advantage, American Society of Mechanical Engineers (ASME)

SELECTED PUBLICATIONS/ CONFERENCE PRESENTATIONS

- Hari. K. Duvvuru, X. Wu, and S. R. Kalidindi, "Calibration of Elastic Localization Tensors to Finite Element Models: Application to Cubic Polycrystals", Journal of Computational Materials Science (in press 2007).
- Hari. K. Duvvuru, M. Knezevic, R. K. Mishra and S. R. Kalidindi, "Application of Microstructure Sensitive Design to FCC Polycrystals", Materials Science Forum Vols. 546-549 (2007) pp. 675-680.
- S. R. Kalidindi, Hari. K. Duvvuru, and M. Knezevic "Spectral Calibration of Crystal Plasticity Models," Acta Materialia, 54 (2006) 1631-1639.
- S. R. Kalidindi and Hari. K. Duvvuru, "Spectral Methods for Capturing Crystallographic Texture Evolution During Large Plastic Strain in Metals," Acta Materialia, 53 (2005)3613-3623.
- S. R. Kalidindi**,** J. Houskamp**,** G. Proust, and Hari. K. Duvvuru, "Microstructure Sensitive Design with First Order Homogenization Theories and Finite Element Codes," Materials Science Forum Vols. 495-497 (2005) pp. 23-29.
- Hari. K. Duvvuru and S. R. Kalidindi, "Crystal Plasticity in Cubic Metals using Spectral Methods", TMS Annual Meeting; February 25-March 1, 2007, Orlando, FL, USA
- Hari. K. Duvvuru and S. R. Kalidindi, "Processing Solutions for Targeted Textures in Polycrystalline Metals," Symposium on Materials Design and Optimization, ASME Design Engineering Technical Conference, Salt Lake City, September 2004.
- Hari. K Duvvuru**,** A. Hossain, and C.H. Jenkins, "Modeling of an Active Seam Antenna," (2003), 4th Gossamer Spacecraft Forum, 44th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Norkfolk, VA.