Brigham Young University
BYU ScholarsArchive

Theses and Dissertations

2009-12-17

Efficient Rotation Algorithms for Texture Evolution

Mark W. Esty
Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Mechanical Engineering Commons

BYU ScholarsArchive Citation

Esty, Mark W., "Efficient Rotation Algorithms for Texture Evolution" (2009). *Theses and Dissertations*. 1985.
https://scholarsarchive.byu.edu/etd/1985

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

Efficient Rotation Algorithms

For Texture Evolution

Mark Willis Esty

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

Master of Science

David T. Fullwood, Chair
Brent L. Adams
Tracy Nelson

Department of Mechanical Engineering

Brigham Young University

April 2010

Copyright © 2010 Mark Willis Esty

All Rights Reserved

ABSTRACT

Efficient Rotation Algorithms
For Texture Evolution

Mark Willis Esty

Department of Mechanical Engineering

Master of Science

Texture evolution is a vital component of many computational tools that link structure, properties and processes of polycrystalline materials. By definition, this evolution process involves the manipulation, via rotation, of points in orientation space. The computational requirements of the current methods being used to rotate crystalline orientations are a significant limiting factor in the drive to merge the texture information of materials into the engineering design process. The goal of this research is to find and implement a practical rotation algorithm that can significantly decrease the computation time required to rotate macroscopic and microscopic crystallographic textures.

Three possible algorithms are considered in an effort to improve the computational efficiency and speed of the rotation process. The first method, which will be referred to as the Gel'fand method, is based on a paper, [1], that suggests a practical application of some of Gel'fand's theories for rotations [2]. The second method, which will be known as the streamline method, is a variation on the Gel'fand method. The third method will be known as the principal orientation method. In this method, orientations in Fourier space are written as linear combinations of points on the convex surface of the microstructure hull to reduce the number of points that must be rotated during each step in the texture evolution process. This thesis will discuss each of these methods, their strengths and weaknesses, and the accuracy of the computational results obtained from their implementation.

Keywords: microstructure sensitive design, MSD, texture, rotations, streamlines, principal orientations, microstructure hull

ACKNOWLEDGMENTS

First and foremost I would like to acknowledge that this thesis is as much, or more, the result of my mother's prayers, support, and faith than it is a result of my own efforts. I would like to thank Dr. Adams for his support and guidance as well as Dr. Nelson for his interesting and enlightening classes. I would also like to thank Sadegh, Jay, Josh, Stewart, and the other members of the group in 165 CB who provided assistance, answers, support, and good company throughout my time in the group. But most importantly I would like to thank Dr Fullwood for his Christ-like patience, encouragement, and guidance throughout this process. I did not make his job easy but he supported me far beyond what I deserved. Thank you.

TABLE OF CONTENTS

LIST OF TABLES vii

LIST OF FIGURES ix

1 Introduction 1

1.1 Microstructure Sensitive Design 2

1.2 Terms, Equations, and Visualizations 4

1.2.1 Anisotropy 4

1.2.2 Texture 5

1.2.3 Euler Space 5

1.2.4 Fundamental Zone 6

1.2.5 Microstructure Function 7

1.2.6 Fourier Space 11

1.2.7 Microstructure Hull 12

1.2.8 Property Closures 13

1.3 Brute Force Rotations 14

2 Gel’fand Transformation Matrix Method 17

2.1 Background 17

2.2 Implementation 22

2.3 Gel'fand Results 24

3 Streamlines 27

3.1 Introduction 27

3.2 Obstacles 30

3.3 Implementation 34

3.3.1 Flowlines 35

3.3.2 One-Step: No Constraints.....36

3.3.3 One-Step: Force Constraint 2.....37

3.4 Streamline Conclusions.....37

4 Principal Orientations.....39

4.1 Background.....39

4.2 Selection of Principal Orientations.....41

4.2.1 Delaunay Selection Method.....42

4.2.2 Fukuda Selection Method.....42

4.3 Recovery.....47

4.4 Principal Orientation Results.....47

4.5 Principal Orientation Conclusions.....52

5 References.....55

LIST OF TABLES

Table 1-1: Comparison of Microstructure Functions.. 10

Table 2-1: Comparison of Rotation Time (Brute Force vs. Gel'fand)................................ 24

Table 3-1: Streamline Data.. 30

Table 4-1: Principal Orientation Rotations.. 51

Table 4-2: Comparison of Rotation Time (Brute Force vs. Principal Orientations) 52

LIST OF FIGURES

Figure 1: MSD Triangle……………………………………………………………………3

Figure 2: Conversion of OIM Data to Bunge-Euler Angles……………………………….....6

Figure 3: Bunge-Euler Angle Rotations…………………..……………………………….6

Figure 4: Visualization of the Fundamental Zone…………………………………………....7

Figure 5: Euler Space / Fourier Space Relationship………………………………………11

Figure 6: Euler Space and Fourier Space with a Convex Hull……………………………13

Figure 7: Property Closures…………………………………………………………………..14

Figure 8: Distribution of Rotated Bin Points………………………………………………25

Figure 9: Orientation Movement—Brute Force vs. Gelfand Method………………………..25

Figure 10: Streamline Paths for Rotations around X, Y, and Z Axes………………………..28

Figure 11: Two Parallel Streamlines……………………………………………………….28

Figure 12: Rotation of a Bin Centroid……………………………………………………..29

Figure 13: Rotation of a Bin around X, Y, and Z Axes……………………………………31

Figure 14: Translation and Rotation of a Bin………………………………………………32

Figure 15: Bin Size vs. 1-1 Mapping Failures……………………………………………...33

Figure 16: Bin Rotation without Dispersion………………………………………………..34

Figure 17: Bin Rotation with Dispersion from Re-binning…………………………………...34

Figure 18: 3,000 Points in the Fundamental Zone…………………………………………40

Figure 19: Orientation Points in Fourier Space……………………………………………40

Figure 20: Principal Orientation Points in Fourier Space………………………………….41

Figure 21: Number of Principal Orientations vs. Original Points………………………….43

Figure 22: Microstructure Hulls (Delaunay and Fukuda)………………………………….45

Figure 23: Number of Principal Orientations vs. Fukuda Tolerance………………………….45

Figure 24: Comparison of Hulls Created Using Different Tolerances………………………..46

Figure 25: Visualization of Brute Force Rotations (Euler and Pole Figures)………………...48

Figure 26: Fourier representation of Euler Crystals……………………………………………49

Figure 27: Pole Figures for Brute Force and Principal Orientation Rotations………………..50

Figure 28: Rotational Paths in Fourier Space..…………………………………………………...51

1 INTRODUCTION

Mechanical design has progressed a great deal since the days when Edison tried over a thousand light bulb filaments before he succeeded. Today, instead of trial and error, engineers and designers usually attempt to model a part or a system many times before they physically produce a prototype. Some of these predictive models are mathematical equations; others are visual representations of the desired product. In recent years a number of computer programs and algorithms have been created to merge visual imagery with the mathematical descriptions of object's properties to give designers a more powerful understanding of a potential product's initial state and its reaction to external forces and environments.

However, it is well known that all models are only approximations to reality. In the creation of any model only certain variables are considered essential and all others are usually approximated to a constant value, or ignored altogether. The choice of which variables to retain is always dependant on the economics of time and money; which in turn are directly related to the sensory and computational capabilities of the user. One of the first concepts taught in science is that whenever a physical object or variable is measured the resulting values are always limited by the resolution of the instrument used. Yet, even when these barriers are overcome, the addition of each new variable to most models usually results in an exponential increase in the model's computational requirements. For this reason, most of the scientific models that form the basis of modern engineering were originally developed with significant approximations (frictionless surfaces, ideal fluids, etc). As sensory equipment, computational power, and theoretical understanding have increased, these models have been altered and expanded to include more variables and create a more accurate description of reality.

In this work we will be working with mathematical, computational, and visual models that attempt to describe the composition of anisotropic engineering materials and predict their properties and behavior when exposed to external stimuli. Partly due to advancements in microscopy, scientific knowledge of material microstructures and processing techniques has increased in recent years. Materials have become increasingly specialized, with more variables being considered in their design and processing. At the same time, advances in finite element programs have provided product designers with the ability to quickly and efficiently model the reaction of a material structure to a variety of external forces and loads. Yet, while these two fields are rapidly expanding along parallel directions, there is a lack of clarity in the interactions between designers and material suppliers; with the majority of the communication being in the form of isotropic and macroscopic descriptions of the supplier's material properties and the designer's performance needs. This presents a vast information gap that will need to be bridged for either industry to achieve their full potential.

1.1 Microstructure Sensitive Design

Microstructure Sensitive Design (MSD) seeks to bridge this gap, specifically it focuses on the interactions between the three main areas of emphasis in material science: microstructure, properties, and processing [3].

With current algorithms and computational tools it is possible to predict and calculate the clockwise path of the triangle in Fig. 1. For example, current theories, algorithms, and databases can predict with reasonable accuracy the microstructure that will result from a given processing technique, the physical properties that are associated with a given microstructure, or the process that produced a given set of properties. However, it is difficult or impractical to work counterclockwise or reverse this process with current computational programs. For example, if given only a set of properties it is almost impossible to determine the corresponding microstructure, or to know what process will produce a desired set of properties.

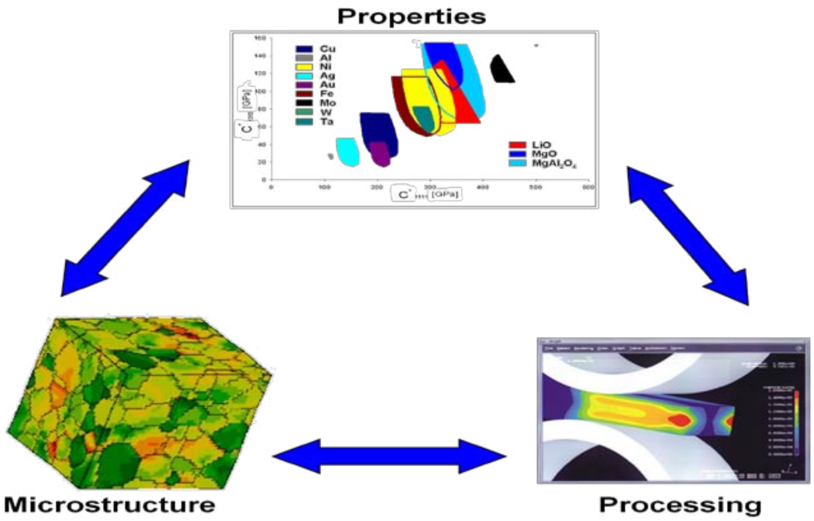


Fig. 1: The interactions between the three main areas of emphasis in materials science are bi-directional but most current analytical approaches are uni-directional

The MSD design teams at BYU and Drexel University are currently working on creating and improving the equations and algorithms that are necessary for a practical computational framework that can model these bi-directional interactions [4-8]. This framework will eventually provide database tools to graphically present all potential material states and properties as well as provide interface capabilities that will allow this data to be incorporated into existing finite element programs and systems. The material information data within this framework will be obtained from a variety of sources; ranging from crystalline orientation and texture information from Orientation Imaging Microscopy scan (OIM) to yield strength information obtained from traditional tensile tests. The MSD program will not only bring all of this material information together in one place, it will transfer the predictive models and equations from textbooks into a practical and unified computer program that can then serve as a bridge between the raw, experimental, material information and the intricacies of the design process.

One major aspect of the MSD computational framework is the viscoelastic behavior of the microstructure of the engineering materials. Every time a polycrystalline material is deformed in a model, each crystalline grain within the sample area is rotated with respect to the surrounding grains and with respect to the sample axis. Thus, with each and every deformation step and corresponding crystalline rotation the entire microstructure function (a mathematical representation of the distribution of crystallographic orientations [4]) must be recomputed. When dealing with data sets that contain a large number of points, and processes that contain numerous deformations, the calculations can quickly become prohibitively expensive. In recent years the application of spectral (Fourier) methods to the MSD framework has significantly reduced the computational load of many of these processes. This work will propose several computational algorithms that may provide additional decreases in computation times and increases in computational efficiency

1.2 **Terms, Equations, and Visualizations**

1.2.1 Anisotropy

In materials science, one of the most common approximations is the assumption of isotropy. To claim isotropy in a material is to assume that the material's parameters (yield strength, toughness, etc) are locally or globally identical in every direction. While this assumption significantly reduces the computation required to model material behavior, it frequently forces the designer to err on the side of safety and produce products that are less efficient in size, weight, or strength than the material's true characteristics might allow. Since the majority of materials are anisotropic (material properties differ depending on direction) this approximation obviously leads to a significant amount of waste and loss of potential in industrial products [7].

1.2.2 Texture

A significant amount of the anisotropy of a macroscopic, physical material can be explained through a detailed analysis of its crystallographic texture. Texture is the distribution of crystallographic orientations in a polycrystalline material [4]. If there is a strong pattern in the orientations the material is highly textured. If the orientations are completely random the material has no texture.

1.2.3 Euler Space

Before crystallographic orientations can be used to define the texture of materials, they must first be obtained and defined relative to a consistent sample frame. For polycrystalline materials one of the best ways to obtain the crystallographic orientation information is through Orientation Imaging Microscopy (OIM) on a Scanning Electron Microscope (SEM)[9, 10]. The orientations obtained from this process (Fig. 2) are in the form of Bunge-Euler angles that represent a set of three rotations (ϕ_1, Φ, ϕ_2) (shown in Fig. 3) that bring the sample frame ($\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$) into coincidence with the local crystal lattice ($\{\mathbf{e}_1^c, \mathbf{e}_2^c, \mathbf{e}_3^c\}$) [5]. These angles can easily be plotted in a three dimensional region know as Euler space. For more information on Bunge-Euler angles and Euler space see [4, 5, 11, 12].

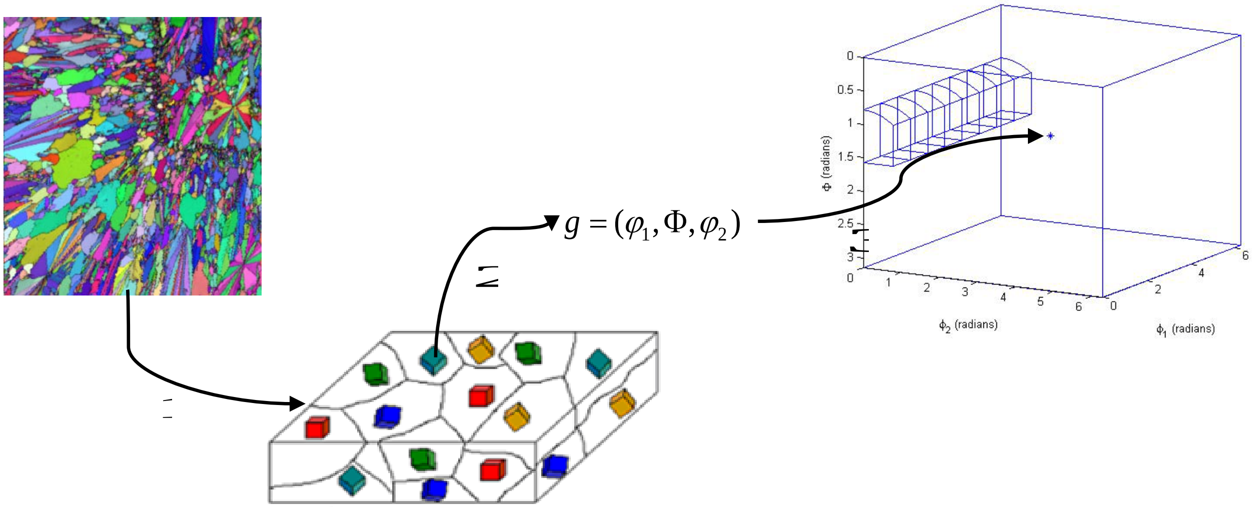

$g = (\varphi_1, \Phi, \varphi_2)$
0
0.5
1
1.5
2
2.5
3
Φ (radians)
0
1
2
3
4
5
6
ϕ_2 (radians)
0
2
4
6
ϕ_1 (radians)

Fig. 2: The crystallographic information at is obtained at a large number of points in a physical sample using OIM (1), this information is then converted into Bunge-Euler angles (2) and can then be easily plotted in three-dimensional Euler Space (3)

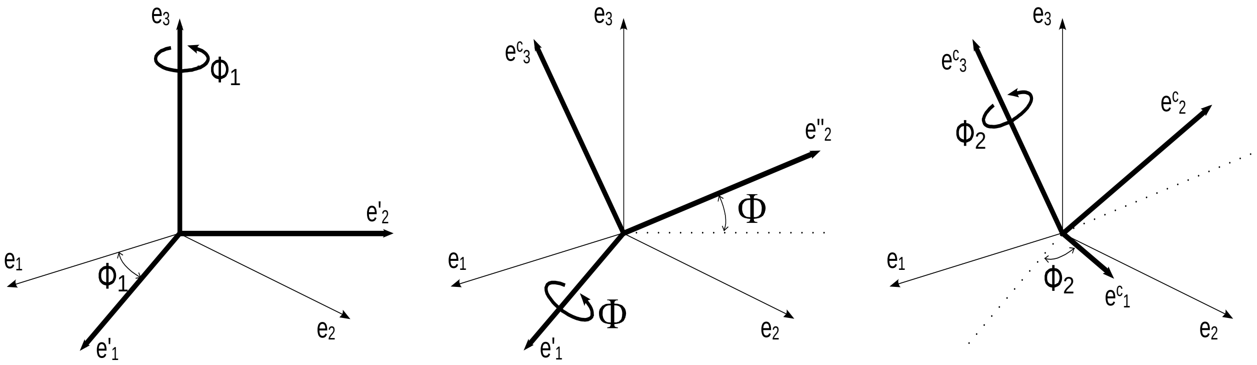
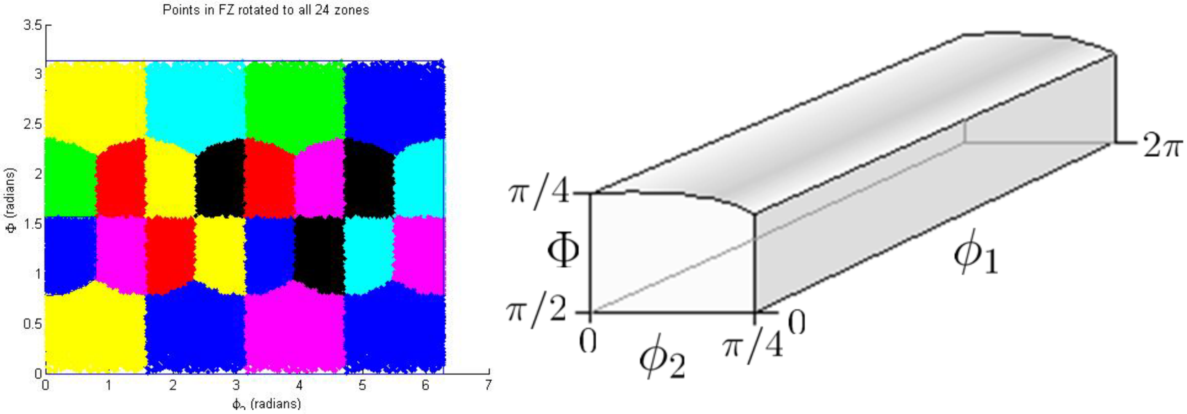

e_3
Φ_1
e'_2
e_1
Φ_1
e'_1
e_2
e_3
e^c_3
e''_2
Φ
e_1
Φ
e'_1
e_2
e_3
e^c_3
e^c_2
Φ_2
e_1
Φ_2
e^c_1
e_2

Fig. 3: Schematic description of the Bunge-Euler angles used to establish the relationship between two arbitrarily defined Cartesian reference frames as a sequence of three rotations [5].

1.2.4 Fundamental Zone

As a result of the symmetry that exists in most crystal structures there may be multiple points inside Euler angle space that describe the same physically-distinct orientation of the local crystal. The smallest volume inside Euler space that can unambiguously describe all possible orientations of a given crystal structure is known as the fundamental zone. For some crystal structures, such as triclinic crystals that possess no symmetry, the fundamental zone occupies all of Euler Angle space, also known as SO(3). In this paper we will deal only with the cubic crystal system (symmetry subgroup C_i) which has 24 different symmetry “zones” (Fig. 4). In this crystal structure the Fundamental Zone (FZ) is commonly defined by the boundaries given in Eq. 1 [5].

$$FZ_C = \left\{ g = (\phi_1, \Phi, \phi_2) \middle| 0 \le \phi_1 < 2\pi, \cos^{-1}\left(\frac{\cos\phi_2}{\sqrt{1+\cos^2\phi_2}} \right) \le \Phi \le \frac{\pi}{2}, 0 \le \phi_2 \le \frac{\pi}{4} \right\} \quad (1)$$



a)

b)

Fig. 4: a)Two-dimensional visualization of the 24 zones of symmetry for the triclinic crystal system b) Three-dimensional visualization of the fundamental zone for the triclinic crystal system [5].

1.2.5 Microstructure Function

In order to understand the manner in which the crystallographic texture of a material affects its macroscopic properties we utilize a mathematical representation of the texture known as a microstructure function. When this microstructure function is limited to 1-point statistics of lattice orientations (i.e. volume fractions of orientation of the crystallographic axes with respect to the macroscopic sample axes, that neglect the form and position of the crystallites [13]) it is often called the Orientation Distribution Function (ODF) [4].

The ODF is generally a continuous function on Euler space. Such a function is difficult to deal with efficiently in computational terms. Hence spectral methods are often employed to discretize the ODF – either by segregating Euler space into a finite number of cells, or by picking out discrete harmonics in the ODF function [14, 15]. Three of the most common spectral methods used to represent the ODF are the Primitive basis, Fourier series based on generalized spherical harmonic functions, and fast Fourier transforms. Each of these representations of the microstructure function has its strengths and weaknesses and thus they are each used in different situations.

The primitive basis is shown in Eq. 2, where F_r is a simple probability / weighting function and $\chi_n(g)$ is essentially a delta function representing an individual crystal orientation bin (if the crystallographic orientation, *g*, lies within bin *n*, the value of this function is 1, otherwise it is zero) [1].

$$f(g) = \sum_{n=1}^{N} F_n \cdot \chi_n(g) \tag{2}$$

In fact, if a discrete representation of Euler space is used, the primitive basis is frequently represented in the form of Eq. 3 where a delta function is used in the place of $\chi_k(g)$ [4, 16, 17].

$$f(g) = \sum_k \alpha_k \, \delta\left(g - g^k\right), \quad 0 \le \alpha_k \le 1 \,, \quad \sum_k \alpha_k = 1 \tag{3}$$

ODF’s that are defined using the primitive basis are very simple to calculate and to represent in visual form. One weakness of this representation is that the computation time required to perform any action on this ODF increases linearly with the number of orientation bins / points that are involved. Other weaknesses of the primitive basis are that it does not retain the symmetry information of the original orientations, it is not a generalized Fourier series (meaning it is not a complete basis and it is not an exact representation of the ODF), and to increase the accuracy (reduce error) you have to refine the entire basis (i.e. decrease the bin size) and then recalculate all of the weighting coefficients.

The second ODF representation we uses Generalized Spherical Harmonics (GSH) as the Fourier basis as shown in Eqs. 4-7 [11].

$$f(g) = \sum_{l=0}^{\infty} \sum_{m=-l}^{+l} \sum_{n=-l}^{+l} F_l^{mn} T_l^{mn}(g) \tag{4}$$

$$F_l^{mn} = (2l+1) \oint f(g) T_l^{*mn}(g) dg \tag{5}$$

$$T_l^{mn}(g) = T_l^{mn}(\varphi_1, \Phi, \varphi_2) = e^{im\varphi_2} P_l^{mn}(\cos\Phi) e^{in\varphi_1} \tag{6}$$

$$P_l^{mn}(\cos(\Phi)) = P_l^{mn}(x) = \frac{(-1)^{l-m} i^{n-m}}{2^l (l-m)!} \left[\frac{(l-m)!(l+n)!}{(l+m)!(l-n)!} \right]^{1/2} \times (1-x)^{-\frac{n-m}{2}} (1+x))^{-\frac{n+m}{2}} \frac{d^{l-n}}{dx^{l-n}} \left[(1-x)^{l-m} (1+x)^{l+m} \right] \tag{7}$$

ODF’s that have been created using the generalized spherical harmonic functions are significantly more powerful than the primitive basis functions since they form a complete Fourier basis. This provides the user with the option of obtaining an exact description of the ODF or truncating the function at a chosen order to increase computationally efficiency. For example, for computations that only involve the elastic deformation only four terms need to be retained [18]. Another valuable benefit of using the GSH functions is that they retain the symmetry information from the original orientations [18]; this is not the case with either the primitive basis or the fast Fourier transforms. However, while the GSH functions contain a great deal of information and can thus be very useful, they can also be very expensive to compute. In practical applications of this series Eqs. 3-6 must be re-computed many, many times and when thousands of orientations are being evolved hundreds of times in succession, the computational cost of computing the P_l^{ml} terms (Eq. 7) becomes particularly imposing.

In order to increase the speed of the process while maintaining some of the benefits of the GSH functions we can turn to discrete Fourier series, calculated using Fast Fourier Transforms (FFTs). Let the three-dimensional Bunge-Euler space of interest be discretized uniformly into $B_1 \times B_2 \times B_3$ bins, and let (b_1, b_2, b_3) enumerate these bins. The DFT representation of the ODF is shown in Eqs. 8 and 9 [3].

$$f(g)=f_{b_1b_2b_3} = \frac{1}{B_1B_2B_3}\sum_{k_1=0}^{B_1-1}\sum_{k_2=0}^{B_2-1}\sum_{k_3=0}^{B_3-1} F_{k_1k_2k_3} e^{2\pi i \frac{k_1b_1}{B_1}} e^{2\pi i \frac{k_2b_2}{B_2}} e^{2\pi i \frac{k_3b_3}{B_3}} \tag{8}$$

$$F_{k_1k_2k_3} = \Im(f) = \sum_{b_1=0}^{B_1-1}\sum_{b_2=0}^{B_2-1}\sum_{b_3=0}^{B_3-1} f_{b_1b_2b_3} e^{-2\pi i \frac{k_1b_1}{B_1}} e^{-2\pi i \frac{k_2b_2}{B_2}} e^{-2\pi i \frac{k_3b_3}{B_3}} \tag{9}$$

Table 1-1: Comparison of Microstructure Functions

	Primitive Basis	**GSH**	**FFT**
Strengths	• Simplicity • Direct connection to orientations and bins	• Retains symmetry information • Is a complete Fourier basis • Important terms are up front allowing for truncation of lesser terms • Directly calculates Fourier coefficients used in MSD tools	• Significantly faster calculations • Can truncate after higher order terms • Directly calculates Fourier coefficients used in MSD tools
Weaknesses	• Large computational requirements • Does not retain symmetry information • Not a complete Fourier basis (not an exact representation of the ODF) • Computationally expensive to increase accuracy	• Computationally expensive • Most complicated terms (P) are frequently recomputed	• Lose symmetry Information • Not a complete Fourier basis

The FFT functions are much simpler to calculate than the GSH functions; note the absence of the T and P functions when comparing Eqs. 8-9 to Eqs. 5-7. Yet they retain the benefits that the GSH functions provided by grouping the most significant terms in the early terms of the summation. As with the GSH functions, the FFT functions also directly produce the coefficients that are required to produce MSD tools such as microstructure hulls and property closures. However, with this significant increase in efficiency there are some sacrifices when using the FFT functions. Since fast Fourier transforms use discrete methods they do not form a complete Fourier basis and the method also does not retain the symmetry information of the orientation points.

1.2.6 Fourier Space

In the same way that each point in real space corresponds to a point in Euler space, every point in Euler space (g) corresponds to a point in Fourier space (F) as a 1-1 mapping (Fig. 5) using equation 10, where T is defined by Eq. 6. Note that this equation is the same as Eq. 5, with the ODF defined by a delta function, for the case of a single crystal.

$$F_l^{mn} = (2l+1)T_l^{*mn}(g) \tag{10}$$

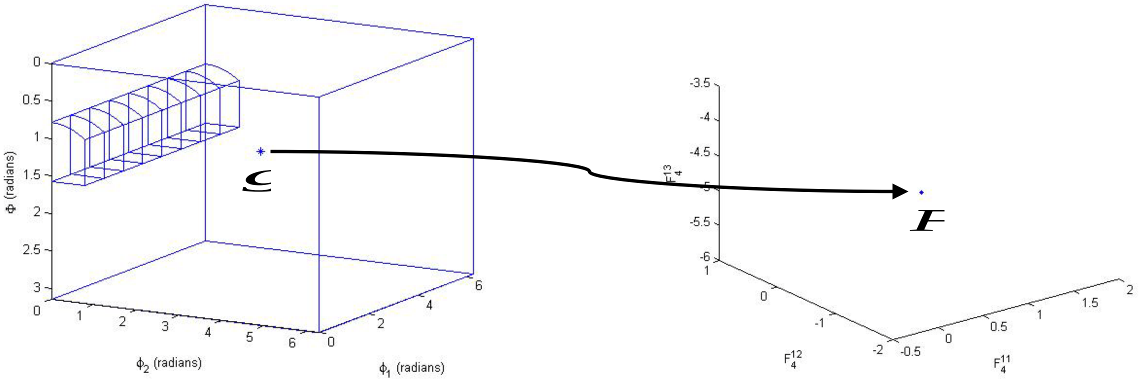


Fig. 5: Representation of the 1-1 relationship between a point in Euler space (g) and the corresponding point in Fourier Space (F), The image of Fourier space is only a three dimensional projection of the full n-dimensional Fourier space

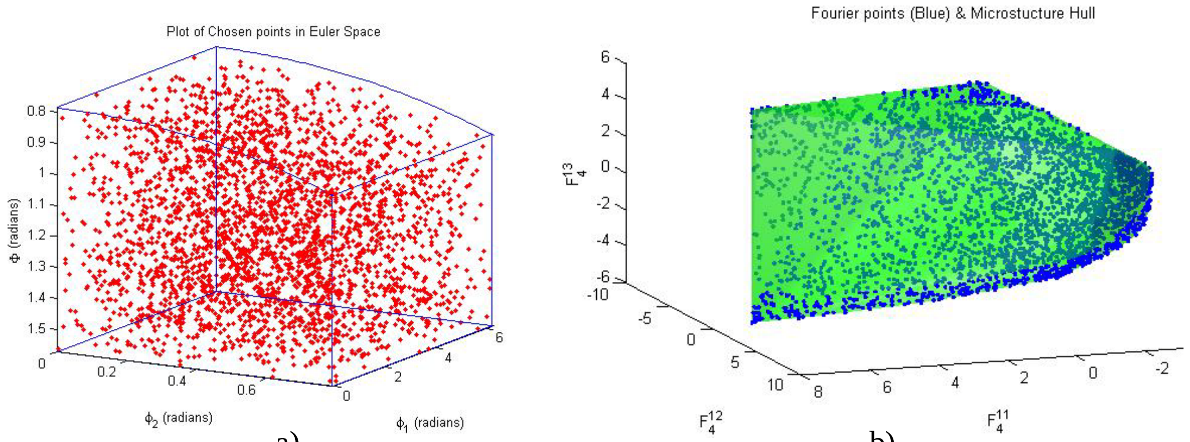
It is important to note that while real space and Euler space are finite in three-dimensions, the full Fourier space contains an infinite number of dimensions. For practical purposes we have used discrete Fourier spaces in our calculations that ranged in size from 3 dimensions up to 24 dimensions. In addition, all images of Fourier space used in this paper are three-dimensional projections of the full Fourier space.

1.2.7 Microstructure Hull

When the full set of all possible orientation distribution functions associated with a particular material or crystal symmetry (Fig. 6-a) has been translated into Fourier space (Fig. 6-b) the filled volume is known as the microstructure hull (*M*) and it represents the complete set of all theoretically feasible ODFs, many of which have not yet been realized in practice or even been targeted for manufacture by materials specialists.[8, 17, 19, 20] This region has been shown to be a convex set. [8, 17, 21] For a set of points to be convex it must contain within its volume all of the line segments that connect any two points in the set [22]. This definition is very pertinent to the principal orientations method that we will discuss towards the end of this report. The mathematical definition of the microstructure hull is dependent on the form of the microstructure function being employed. Equations 11 and 12 define the GSH form of the microstructure hull [16, 17] and this is the mathematical definition that is used in the principal orientations method discussed in chapter 4.

$$M = \left\{ F_l^{mn} \middle| F_l^{mn} = \sum_k \alpha_k \, {}^kF_l^{mn},\ {}^kF_l^{mn} \in M^k, \alpha_k \geq 0\ , \sum_k \alpha_k = 1 \right\} \tag{11}$$

$$M^k = \left\{ {}^kF_l^{mn} \middle| {}^kF_l^{mn} = \frac{1}{(2l+1)} T_l^{*mn}\left(g^k\right), g^k \in FZ \right\} \tag{12}$$



a)   b)

Fig. 6: A set of 3,000 orientations that represent the region of all possible orientations for cubic materials: shown as points in Euler space (a) and as points in Fourier space enclosed by a convex hull (b).

1.2.8 Property Closures

These same coefficients can then be used to calculate the physical properties associated with their corresponding orientations. For example, equation 13 produces all feasible combinations of macroscale elastic properties for the material defined by Fourier coefficients $F_l^{\mu\nu}$[17].

$$\langle S_{abcd}\rangle = \oint S_{abcd}(g)\, f(g)\, dg = \sum_{l=0}^{4}\sum_{m=1}^{N(l)}\sum_{n=1}^{M(l)} \frac{1}{(2l+1)}\; {}_{abcd}S_l^{*mn} F_l^{mn} \qquad (13)$$

Once the material properties associated with each point in Fourier space have been calculated we can combine them to create a multidimensional property closure that delineates the complete set of theoretically feasible effective (homogenized) anisotropic property combinations in a given material system (Fig. 7) [17].

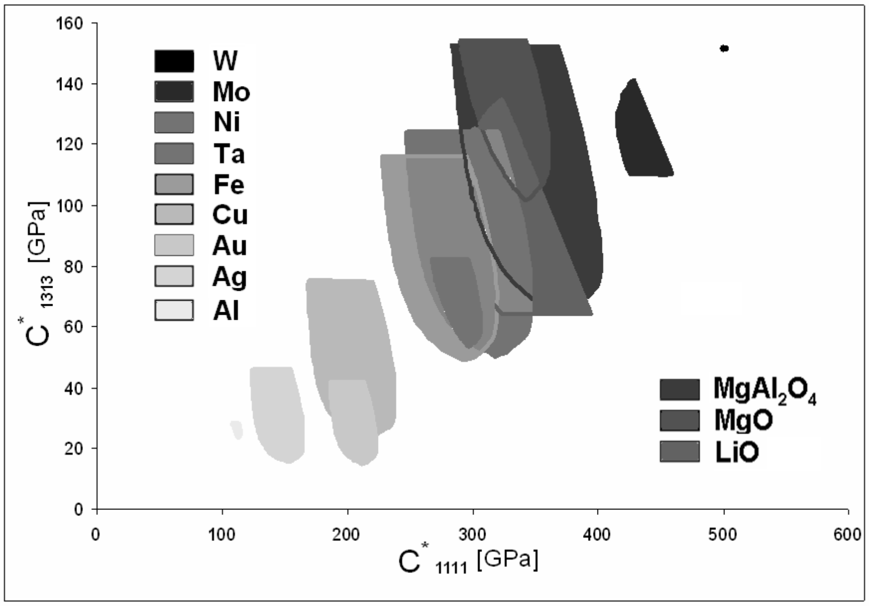


Figure 7: Atlas of $\left(C^*_{1111}, C^*_{1313}\right)$ property closures for a broad selection of cubic materials [17]

1.3 Brute Force Rotations

Depending upon the representation of the ODF being employed, evolution of the ODF must determine the effect of applied rotations upon the original Euler angles, or upon the spectral coefficients of the ODF. The simplest and most fundamental method of texture rotation is to individually rotate each of the crystal orientation data points directly from their Bunge-Euler angle form. This process involves converting each set of Euler angles, $[\varphi_1, \Phi, \varphi_2]$, to a 3x3 orientation matrix (Eq. 14). [11] It is important to note that equation 14 is the active form of the equation presented in passive form by Bunge in [11]. This matrix can then rotated around the desired reference frame using equations 15-17; where equation 15 is the matrix for rotations around the Z-axis, equation 16 is used for rotations around the X-axis, and equation 17 is used for rotations around the Y-axis.[11]

$$g(\varphi_1,\Phi,\varphi_2)=\begin{bmatrix}\cos(\varphi_1)\cos(\varphi_2)-\sin(\varphi_1)\sin(\varphi_2)\cos(\Phi) & -\cos(\varphi_1)\sin(\varphi_2)-\sin(\varphi_1)\cos(\varphi_2)\cos(\Phi) & \sin(\varphi_1)\sin(\Phi)\\ \sin(\varphi_1)\cos(\varphi_2)+\cos(\varphi_1)\sin(\varphi_2)\cos(\Phi) & -\sin(\varphi_1)\sin(\varphi_2)+\cos(\varphi_1)\cos(\varphi_2)\cos(\Phi) & -\cos(\varphi_1)\sin(\Phi)\\ \sin(\varphi_2)\sin(\Phi) & \cos(\varphi_2)\sin(\Phi) & \cos(\Phi)\end{bmatrix} \tag{14}$$

$$\begin{bmatrix}\cos(\theta) & -\sin(\theta) & 0\\ \sin(\theta) & \cos(\theta) & 0\\ 0 & 0 & 1\end{bmatrix} \tag{15}$$

$$\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(\theta) & -\sin(\theta)\\ 0 & \sin(\theta) & \cos(\theta)\end{bmatrix} \tag{16}$$

$$\begin{bmatrix}\cos(\theta) & 0 & \sin(\theta)\\ 0 & 1 & 0\\ -\sin(\theta) & 0 & \cos(\theta)\end{bmatrix} \tag{17}$$

After rotation, the Bunge-Euler angles are retrieved from equation 14 using equation 18.

$$\begin{aligned}\varphi_1 &= \tan^{-1}\left(\frac{\sin(\varphi_1)\sin(\Phi)}{-\cos(\varphi_1)\sin(\Phi)}\right)\\ \varphi_2 &= \tan^{-1}\left(\frac{\sin(\varphi_2)\sin(\Phi)}{\cos(\varphi_2)\sin(\Phi)}\right)\\ \Phi &= \cos^{-1}(\cos(\Phi))\end{aligned} \tag{18}$$

The special cases of $\Phi = 0$ and $\Phi = \pi$ are directly handled in the code, but there remains an ambiguity in the retrieval algorithm when a point is rotated out of Euler angle space in the positive or negative Φ direction (increasing past π or decreasing below zero). In this paper we will refer to the rotation algorithm described above as the Brute Force method of rotation.

The Brute Force method, while relatively simple, can carry an extremely expensive computational price when rotating large data sets that are in ODF form. When using the brute force method to rotate an ODF, the microstructure functions must first be returned to their original Bunge-Euler angle form. These angles are then converted into the Bunge matrix found in equation 14. Only then can the rotation, g', be applied to the matrix to produce the new crystalline orientations, g''. The angles are then returned to Bunge-Euler angle form. The spectral form of the new ODF can be found through the re-computation of either the coefficients ($\boldsymbol{P}_l$, $\boldsymbol{F}_l^{mn}$, $\boldsymbol{F}_{k_1k_2k_3}$) or the GSH functions ($T_l^{mn}(g)$). Most previous rotation algorithms in the MSD group have focused on re-computing the GSH functions.

When the rotation being performed, g', is contained within a set of pre-computed rotations, g, the addition theorem (Eq. 19)

$$T_l^{mn}(g \cdot g') = \sum_{s=-l}^{+l} T_l^{ms}(g) T_l^{sn}(g') \tag{19}$$

can be used to obtain the new T functions more efficiently. Since this scenario is not that common, it frequently becomes necessary to recalculate Eq. 6 where Eq. 7 is easily the most expensive computational process. Using the small angle approximation of $\cos\Phi \cong 1$ in Eq. 6 can also have a positive impact on computation time, but at the expense of accuracy if repeatedly employed during a structure evolution. However, neither of these adjustments can change the overall nature of the brute force rotation algorithm, which has a linear or quadratic relationship between the size of the input data and the computation time.

2 GEL’FAND TRANSFORMATION MATRIX METHOD

The first method that we will present uses small-angle approximations of axis angle parameters to create a transformation matrix that can operate on the coefficients of the ODF and rotate the texture in constant-time (i.e. the length of time required for the rotation is independent of the size of the data). The creation and application of this transformation matrix was suggested in [1] and follows previous work on Lie Groups described by Gel’fand, Minlos, and Shapiro.[2] Indeed, most of the background and implementation sections in this chapter are taken directly from [1]. This chapter will demonstrate this Gel'fand method using the primitive basis representation for the ODF that is discussed above (Eq. 2)

2.1 **Background**

As mentioned above, each coefficient in Eq. 2 contains the volume fraction of crystalline orientation measurements, g , from the sample that are associated with a small volume in Euler angle space, dg, that we will refer to as a bin, ω_r. These bins are created by partitioning the fundamental zone into N smaller volume elements, $\omega_1, \omega_2, ..., \omega_n, ..., \omega_N$ where the properties of each bin are defined by Eq. 20 ($\varnothing$ is the empty set.):

$$\bigcup_{n=1}^{N} \omega_n = FZ, \quad \omega_i \cap \omega_j = \varnothing \; (i \neq j, \; i, j = 1, 2, ..., N) \qquad (20)$$

The dimensions of each bin are defined such that their measure, $m(\omega_i)$, is equivalent to every other bin (Eq. 21)

$$m(\omega_i) = \iiint_{\omega_i} dg = 1/N \tag{21}$$

Here dg is the invariant measure on the orientation space [2, 18, 23]. Summing Eq. 21 over the entire set of orientation bins (Eq. 22) defines the total measure of the *FZ* to be 1.

$$\sum_{n=1}^{N} m(\omega_n) = 1 \tag{22}$$

These bins are then used to define the functions in Eq. 2, $\chi_n(G)\ (n = 1, 2, \ldots, N)$, in the form of Eq. 23.

$$\chi_n(G) = \begin{cases} 1 \ \mathit{if}\ \ G \in \omega_n \\ 0 \ \mathit{otherwise} \end{cases} . \tag{23}$$

The transformed (rotated) ODF, $Rf(g)$, which is the volume-fraction density of lattice orientations in the twisted sample can be related to the original ODF, $f(g)$, through Equation 24:

$$Rf(gR) = f(g) \ . \tag{24}$$

Thus, if $f(g)$ is known, and *R* is fixed, $Rf(g)$ can be determined from Eq. 24. However, instead of working from this direction, we will be using Eq. 25 where the rotation acts directly on the coefficients in the series.

$$Rf(g) = \sum_{n=1}^{N} RF_n \chi_n(g) \tag{25}$$

Since the coefficients in both Eqs. 2 and 25 are volume fractions it follows that the sum of the new rotated coefficients will be identical to the sum of the original coefficients (Eq. 26).

$$\sum_{n=1}^{N} F_n = N = \sum_{n=1}^{N} RF_n \qquad (26)$$

One of the benefits of working with rotations in the form of Eq. 25 lies in the fact that the rotation operates on the coefficients rather than on the crystalline orientations themselves. This is valuable for several reasons. First, the number of coefficients in the series is dictated by the number of bins the FZ has been divided into. Since the number of bins is usually significantly less than the number of measured orientation points, this difference alone can drastically decrease the number of calculations that must be done. In addition, the number of bins will likely remain constant while the number of orientation points in an ODF can vary widely, thus the calculation time will become less variable. Second, operating directly on the coefficients automatically removes the necessity of re-calculating these coefficients after each rotation, providing further savings on computation time and resources. Third, since the number and bin assignments of the coefficients does not change for different materials, we have the opportunity to create a transformation / rotation matrix that can be calculated once for a given set of bins and then used for all ODFs that have the same crystal symmetry.

To operate directly on the coefficients an $N \times N$ transformation matrix $\boldsymbol{T}_{I}$ is defined that transforms the coefficients of the ODF, $\boldsymbol{F}_r$, into those of the rotated ODF, $\boldsymbol{RF}_n$, upon twisting the sample by a rotation *R*. This transformation is expressed in Eq. 27.

$$\begin{bmatrix} RF_1 \\ \vdots \\ RF_m \\ \vdots \\ RF_N \end{bmatrix} = \begin{bmatrix} T_{R11} & \cdots & T_{R1N} \\ \vdots & & \vdots \\ T_{RN1} & \cdots & T_{RNN} \end{bmatrix} \begin{bmatrix} F_1 \\ \vdots \\ F_n \\ \vdots \\ F_N \end{bmatrix} \tag{27}$$

The individual components, T_{Rmn}, describe the fraction of invariant volume belonging to bin ω_n that transfers into bin ω_m. When multiplied by F_n, T_{Rmn} signifies the fraction of the initial ODF that contributes to the component RF_m of the new ODF. Thus, if no rotation has taken place the diagonal components, T_{Rnn}, of the matrix T_R will all be equal to 1, and if 10% of a bin's points have been rotated out of the bin then T_{Rnn} should equal 0.9 and the rest of the components in that row should add up to 0.1. This definition of T_R leads naturally to two 'conservation of mass' relationships (Eq. 28 and Eq. 29) that should be satisfied by the coefficients of T_R.

$$\sum_{m=1}^{N} T_{Rmn} = 1 \ \text{(for all } n = 1, 2, ..., N) \tag{28}$$

$$\sum_{n=1}^{N} T_{Rmn} = 1 \ \text{(for all } m = 1, 2, ..., N) \tag{29}$$

In practice, these equations mean that when a random ODF has been created (i.e. all the bins have an equal, or nearly equal, number of orientation points in them) any rotation that operates uniformly on all the bins should move as many points into a given bin as are rotated out of it (Eq. 29: the columns of T_{Rmn} should sum to 1). In addition, any point that is rotated out of a given bin must enter another of the bins (Eq. 28: the rows of T_{Rmn} should sum to 1). If R is an infinitesimally small rotation then T_{Rnn} will be much larger than the other terms. Also, due to the nature of the rotations and the definition of T_R it must be true that T_{Rmn} can never be larger than one or smaller than zero (Eq. 30).

$$0 \le T_{Rmn} \le 1 \ (\mathit{for\ all}\ m, n = 1, 2, \ldots, N) \tag{30}$$

That T_R forms a representation of the N-dimensional real space (in this case, the space of N-dimensional approximations of the ODF) is evident from two basic properties of the transformations. The first is:

$$T_{R_1} T_{R_2} = T_{R_1 R_2} \tag{31}$$

The second property is the presence of the identity transformation, T_e, which is the NxN dimensional identity matrix:

$$[T_e] \equiv \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & \ddots & 0 & \cdots & 0 \\ \vdots & 0 & 1 & 0 & \vdots \\ 0 & \vdots & 0 & \ddots & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \tag{32}$$

Clearly the effect of T_e is to leave the ODF unchanged. The properties expressed in Eqs. 31 and 32 define T_R to be a representation of the rotation group [2].

Following the traditional approach of Lie group theory [2], it is convenient to parameterize the rotations that twist the sample using the axis-angle parameters. Let $\vec{\xi} = \xi_1 \hat{e}_1^S + \xi_2 \hat{e}_2^S + \xi_3 \hat{e}_3^S$ represent the rotation $\xi = |\vec{\xi}|$ in a right-handed sense about an axis $\hat{n} = \vec{\xi} / |\vec{\xi}|$. The magnitude of the rotation is restricted to the range $0 \le \xi \le \pi$. Thus, the set of all rotations belongs to the □-sphere [2, 23]. In this approach, $T_R = T_R(\xi_1, \xi_2, \xi_3)$, and elements of the transformation matrix, $T_{Rmn} = T_{Rmn}(\xi_1, \xi_2, \xi_3)$, are continuous functions of the rotation variables

ξ_1,ξ_2,ξ_3. Hereafter we will shorten the notation for the transformation matrix to

$T_R(\xi_1,\xi_2,\xi_3) = T_{\vec{\xi}} = T(\xi_1,\xi_2,\xi_3)$.

2.2 **Implementation**

When considering the implementation of the transformation matrices it should be remembered that the components of the transformation matrix are dependent upon the particular processes and choices used when partitioning the fundamental zone, *FZ*, into bins ω_n. However, the relatively simple procedure applied below can be used with any particular choice of binning to determine the components of $T_{\vec{\xi}}$ for any choice of rotation step size, $\vec{\xi}$. The *FZ* was first filled randomly with a large number of points in the Bunge-Euler angle form, $g = (\phi_1, \Phi, \phi_2)$, that represented a random distribution of orientations. Given that the invariant measure for each ω_n is equivalent in our procedure, the number of random points found in each bin should be consistent. The sample twist is then chosen and each orientation point is changed according to $g \to g\vec{\xi}$. If the number of random points found in ω_n, before twisting, is C_r, and if $B_m^n(\vec{\xi})$ of those points are found in bin ω_m after the application of *R*, then $T_{\vec{\xi}m}$ is defined by Eq. 33.

$$T_{\vec{\xi}mn} \approx \frac{B_m^n(\vec{\xi})}{C_n} \tag{33}$$

For most of the implementations shown below we used 1,000,000 random points and 10,890 bins which corresponds to an average bin edge dimension of 4° and an average number of points per bin of C_r= 918±134. For very small rotations, ξ_1,ξ_2,ξ_3 can be treated as rotations around the x, y, and z axes respectively. Therefore we calculated a separate T_R function for rotations of 0.001 radians (0.0573˚) around each of these axes. One of the fundamental ideas behind using these transformation matrices is that any large rotation can be represented by a summation of much smaller rotations. To accomplish this we used Taylor’s theorem (Eq. 34) to calculate the total transformation matrix from the $\mathrm{T}_{\vec{\xi}\mathrm{m}\imath}$ matrix above that represents a very small rotational step.

$$T(\xi_1, \xi_2, \xi_3) = T_e + A_1\xi_1 + A_2\xi_2 + A_3\xi_3 + \ldots \tag{34}$$

Since the higher order terms in Eq. 34 will be negligible in comparison with $\sqrt{\xi_1^2 + \xi_2^2 + \xi_3^2}$, the +... indicates that we have omitted these terms in the Taylor’s series.

Recalling that $e^x = 1 + x + \frac{1}{2!}x^2 + \ldots$, equation 34 can be rewritten in the form of Eq. 35 [2].

$$T(\xi_1, \xi_2, \xi_3) = e^{A_1\xi_1 + A_2\xi_2 + A_3\xi_3} \tag{35}$$

The coefficients (*A₁*, *A₂*, and *A₃*) in the Taylor's expansion above are referred to as infinitesimal rotation matrices. These matrices are defined in Eq. 36.

$$A_1 = \left.\frac{\partial T(\xi_1, 0, 0)}{\partial \xi_1}\right|_{\xi_1=\xi_2=\xi_3=0},\ A_2 = \left.\frac{\partial T(0, \xi_2, 0)}{\partial \xi_2}\right|_{\xi_1=\xi_2=\xi_3=0},\ A_3 = \left.\frac{\partial T(0, 0, \xi_3)}{\partial \xi_3}\right|_{\xi_1=\xi_2=\xi_3=0} \tag{36}$$

The relationship between these matrices is described by Gelfand, et al [2], with the most significant condition being commutation relations that interconnect them (Eq. 37).

$$[A_1, A_2] = A_3,\ [A_2, A_3] = A_1,\ [A_3, A_1] = A_2 \tag{37}$$

where $[A, B] = AB - BA$.

2.3 Gel'fand Results

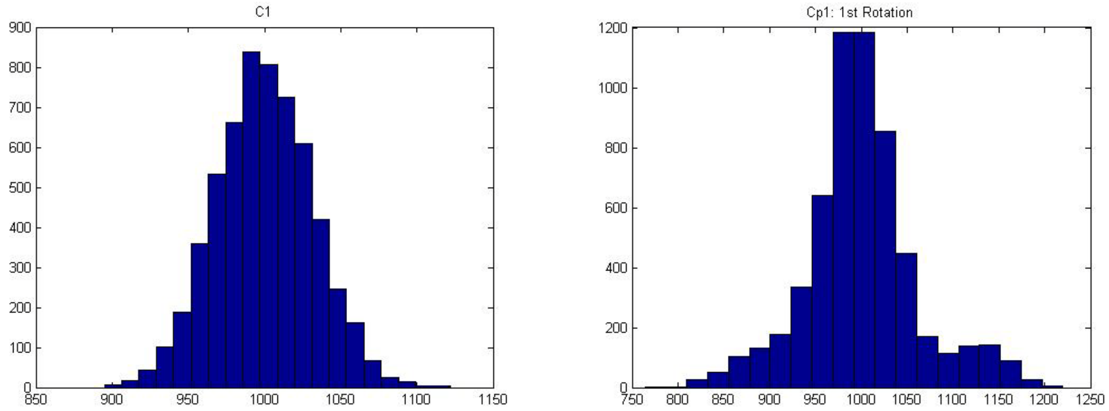
After calculating the transformation matrices as described above, we used them to rotate a simple, known ODF and compared these results to an identical rotation computed using the brute force method. The first result we found was that for large quantities of crystalline points the constant-time nature of the Gel'fand method calculates the new rotated ODF significantly faster.

Table 2-1: Comparison of Rotation Time (Brute Force vs. Gel’fand)

Points	Time (Brute Force)	Time (Gel’fand)
10,000	14 sec	475 sec
10,000,000	1,050 sec	378 sec

However, speed is only beneficial if the accuracy of the rotation process is maintained, and in this area the practical implementation of the Gelfand transformations-matrix method falls short of expectations. It appears that most of the inaccuracies that arise can be traced back to the inherent loss of accuracy when applying a continuous mathematical formulation to a discontinuous, discrete implementation.

For example, while the rotation size and angle are constant for all crystal orientations their displacement within the fundamental zone in not entirely uniform. Thus, though all the bins start with nearly similar volume fractions (Fig. 8-a), after the rotation the volume fractions have decreased in uniformity (Fig. 8-b). This quirk of Euler space causes the requirement in Eq. 28 (that all bins have the same invariant measure before and after rotation) to not be met. Eq. 29 still remains computationally valid since all point in each bin either stayed in that bin or were rotated into a new bin.



a) b)

Fig. 8: Distribution of points among bins before rotation (a) and after rotation (b) (x-axis is the number of points in a bin and the y-axis is the number of bins containing that number of points)

In addition, the discrete nature of the bins significantly increases the speed of the point dispersion. Fig. 9 shows how the volume fraction of points in the individual bins decreases much faster using the Gel'fand method of rotation. It does not take long for the Gel'fand method to produce a smooth distribution from the original, compact, discrete ODF. In contrast the Brute Force method retains the shape of the ODF and consistently moves the discrete volume fraction from bin to bin, only losing a small percentage to neighboring bins—this is one of the things that inspired the streamline method that we will discuss below.

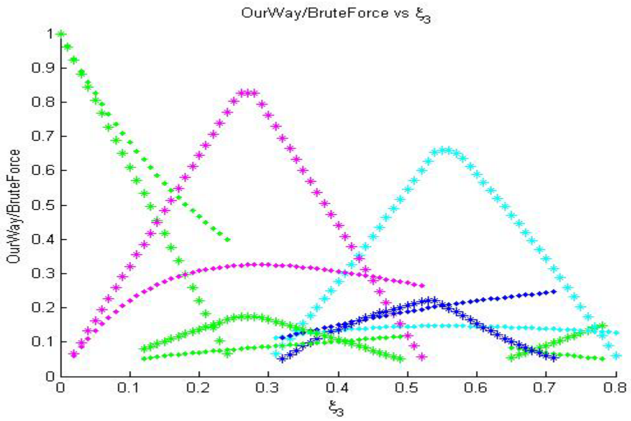


Fig. 9: Comparison of orientation movement during a brute force rotation (‘*’, peaked paths) and Gel'fand rotation (‘.’, smooth paths)

In an attempt to ensure that we were using the proper parameters in the creation of our transformation matrices, we systematically altered several input variables to find where their impact on the variability of the results flattened out. We determined that small changes in the rotation size had almost no effect on the results, that the number of random points created in Euler space should be above 800,000, and that the bin size does have a significant effect.

Despite all its promise, until computational power can increase to the point where bin sizes are small enough to approximate a continuous function, the Gel'fand rotation algorithm will be incapable of providing reliable rotations.

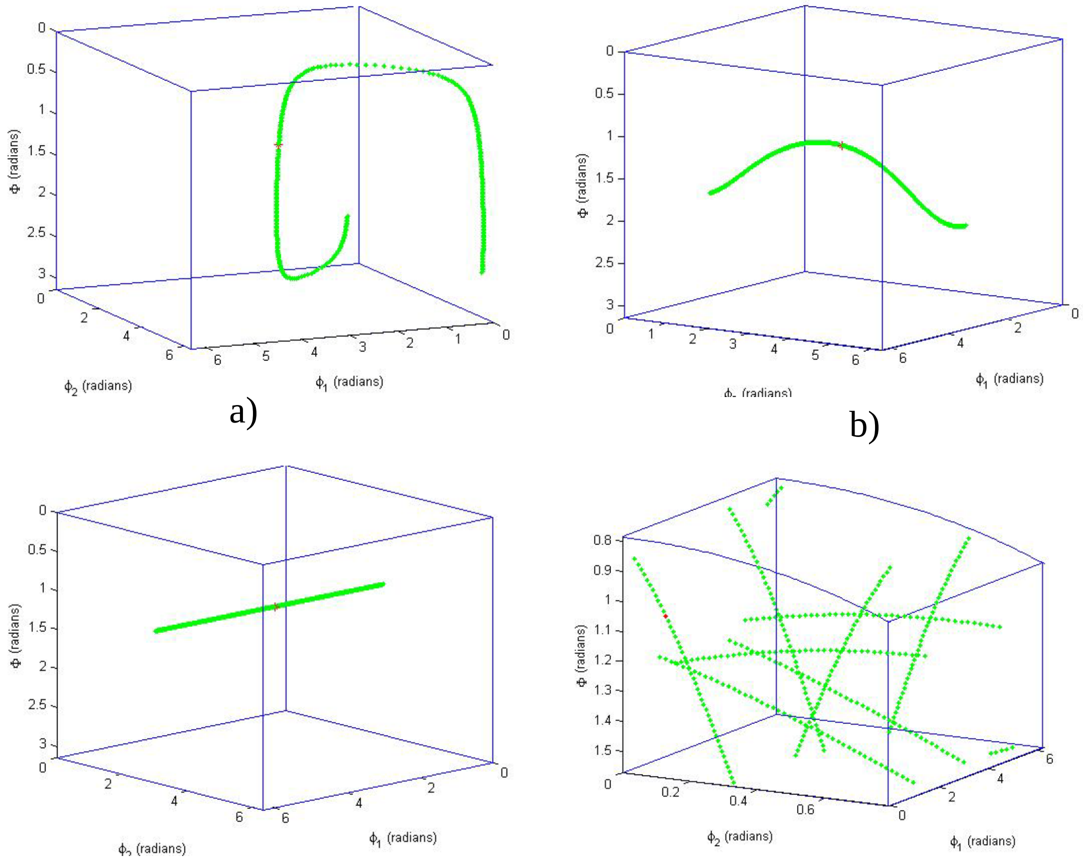
3 STREAMLINES

3.1 Introduction

The practical implementation of the Gel’fand method discussed above brings attention to the idea that upon rotation around a given axis (x, y, or z) each point will follow a unique and repeatable path through Euler space or the fundamental zone (Fig. 10). From the periodic nature of Euler angles it is clear that any orientation in Euler space will return to its original location after being rotated 360° (Fig. 10). In addition, if we assume infinitesimal points, g_i, such that for all g_i Eq. 38 is satisfied, then the rotational paths associated with any two points, $\{g_i, g_k: i \neq k\}$ cannot cross (show in Fig. 11). We will refer to these rotational paths as streamlines.

$$\begin{aligned} g_i \neq g_k \therefore i \neq k \\ g_i = g_k \therefore i = k \end{aligned} \tag{38}$$

Since the streamline associated with a given orientation and rotation direction is independent of the rotation step size, once a streamlines has been calculated for a given point any rotation acting on this point that is larger than the step size can be found by simply moving the appropriate distance along the streamline path. This characteristic of streamlines suggests that all possible rotations on every point in the fundamental zone can be pre-computed and stored in “look-up tables” that could provide constant-time access to crystallographic rotation information.



a) b)

c) d)

Fig. 10: Representation of a point (*-red) in Euler as it is rotated 360° around the x-axis (a) y-axis (b) and z-axis (c) using 1° steps. Figure (d) shows the streamline points in 'a' after they have been rotated into the fundamental zone

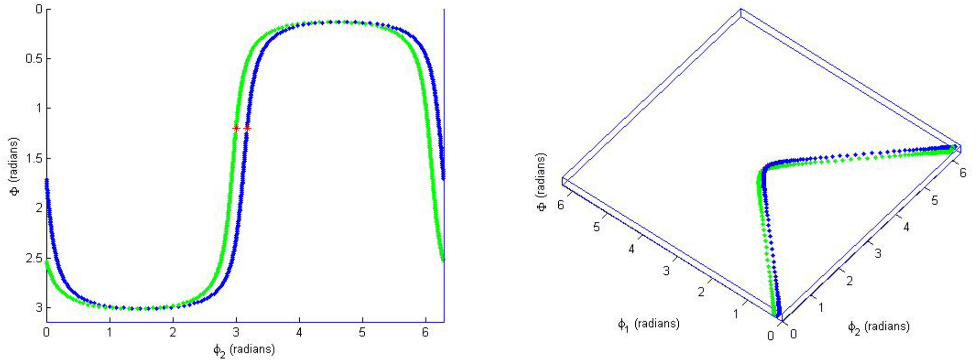


Fig. 11: Two streamlines that begin (*-red) 10° apart before being rotated 360° in 1° increments (shown from two angles to demonstrate that the streamlines do not cross)

However, as in the Gelfand method above, the practical implementation of this theory requires the discretization of the fundamental zone into individual volume elements or bins, ω_i. If the bins are bins are defined like those is Chapter 2 (Eqs. 20-22), then it can be assumed that the streamlines associated these bins obey the same rules as those associate with individual orientation points (i.e. they form closed loops and they do not cross). This suggests that the bin streamlines can be reasonably approximated by the average of the streamlines associated with the each of the individual points within the bin volume. Fig. 12 shows one such approximation that uses the streamline for the centroid of a bin to approximate a streamline for the entire bin volume.

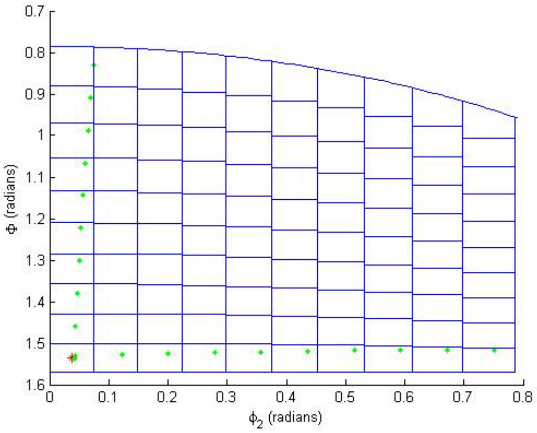


Fig. 12: Representation of a streamline beginning at the bin centroid in the lower left hand bin

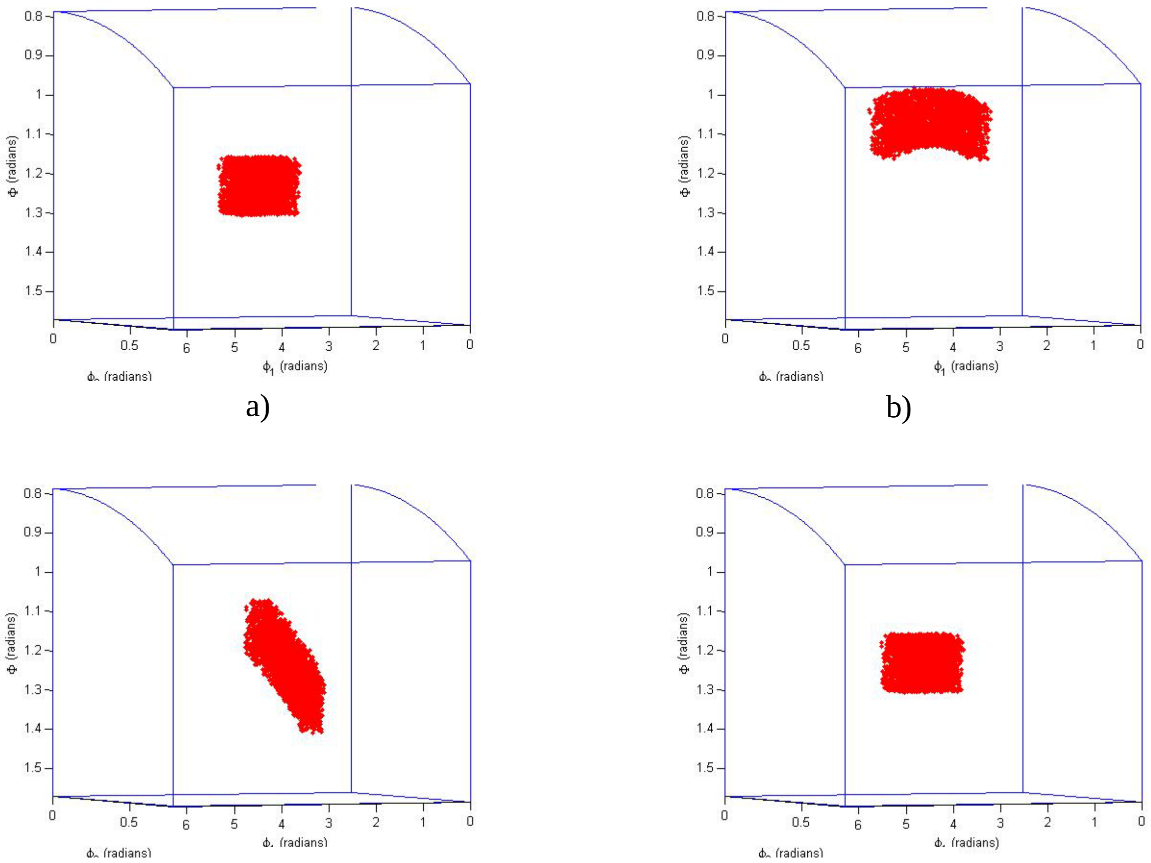
The basic concept behind this method is that when the rotational step size is the same as the average bin dimension, or larger, then each rotational step will assign the volume fraction associated with the original bin to a new bin. A simplistic demenstration of this concept is shown in table 3.. This table represents a partial and condesed streamline data set with an average bin dimension of 15° and a rotation step size of 60° around the x-axis. The left-most column represents the original bin location with each following column representing the new bin location after a rotation of 60° For example, using table 3, if 40% of the ODF was originally in bin 7 and it was rotated by 60° we can use row 7 to determine that this volume fraction should now lie in bin 1. If we then impose an additional rotation of 120° we will use row 1 and find that this volume fraction should lie in bin 49.

Table 3-1: Streamline Data

1	25	49	151	79	7	1
2	26	50	62	38	8	2
3	93	189	205	181	157	3
4	94	190	206	182	158	4
5	23	47	71	5	23	5
6	24	48	72	150	78	6
7	1	25	49	151	79	7
8	2	26	50	62	38	8
9	21	117	213	197	126	9
10	22	118	214	198	127	10

3.2 **Obstacles**

However, Fig. 12 also demonstrates one of the major issues involved in using rectangular bins to represent and record the streamline paths. It is readily apparent that the centroid point of the starting bin does not follow / match the centroid points of the bins it enters, and actually gets increasingly off-center as the first 9 rotations (at 4.5° per rotation) progress before it begins to return to the $\Phi - \varphi_2$ centroid position (note that this is a two-dimensional representation of a three-dimensional rotation and so bin orientation in the φ_1-direction is not shown). The difference between the shape of the original bins and the shapes of the bins after rotation is shown in more detail in Fig. 13. From these figures it is clear that any rotation around the z-axis maintains the shape of the bin and is therefore a pure translation in the φ_1-direction (Fig. 13-d), whereas rotations around the y-axis and the x-axis significantly distort the shape of the bins.



a) b)

c) d)

Fig. 13: a) a filled rectangular bin that has not been rotated; b) the bin in `a' after being rotated 10° around the x-axis; c) the bin in `a' after being rotated 10° around the y-axis; d) the bin in `a' after being rotated 10° around the z-axis

This distortion does not interfere with the ability of the volume as a whole to form a streamline, but it does cause significant problems in trying to assign the streamline points to the bins they intersect. Invariably, no matter how the points used to identify the bin are chosen, there will be be portions of the bin volume that lie outside the new bin to which they are assigned. In addition, we have found that upon rotation the centroids of two bins will frequently enter one bin, leaving another (usually somewhat distant) bin empty. This is likely due to the rotational effect that is shown, in a very rough form, in figure 13. When an object or group of objects are rotated by an equal amount the actual distance traveled by any one point in orientation space is dependant on its distance from the rotational axis. This could possibly be adjusted for by using polar coordinates to bin the fundamental zone, but the convaluted geometry of the x and y rotational axis and the invariant meaure in the Φ direction preclude this as a viable option.

This “clumping” of the bins could also be an effect of the asymetrical “loaf” shape of the fundamental zone that forces the bins to have different shapes to hold their identical invarient volumes.

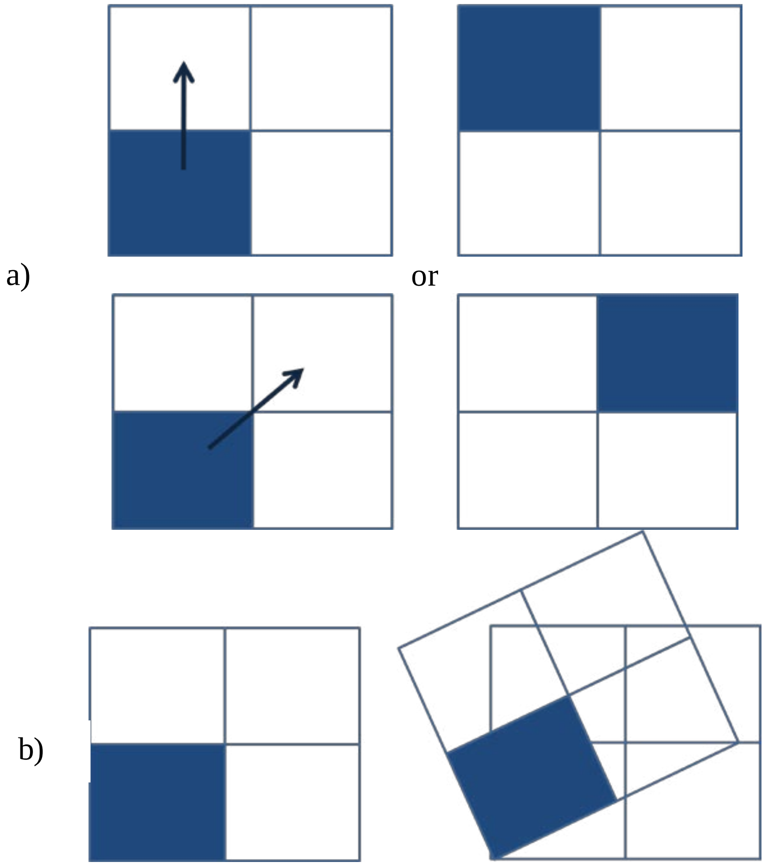


Fig. 14: a) pure translation of a bin; b) rotation of a bin

The frequency of this "clumping" effect is shown by the graphs in Fig. 15, where we have discretized the fundamental zone into bins of an average edge-length of θ and then rotated all the bins by one θ-step around around the x-axis. By varying θ between 1° and 20° and measuring the number of errors (defined as the number of empty/unassigned bins) at each bin size we can see that while the number of errors increases with decreasing bin size (Fig. 14a), the ratio of bin failures to streamlines significantly decreases (Fig 14b). Thus, the accuracy of the streamline functions should increase as the bin size decreases and the number of bins increases. The computational lower limit on bin size is determined by the storage parameters of the computer hardware and programming language being used. Matlab cannot store 2 dimensional matrices larger than 800,000x800,000 and vectors longer than 800,000, which corresponds to an average bin size of 1°, and at this scale the computational cost in time and computing power is significant.

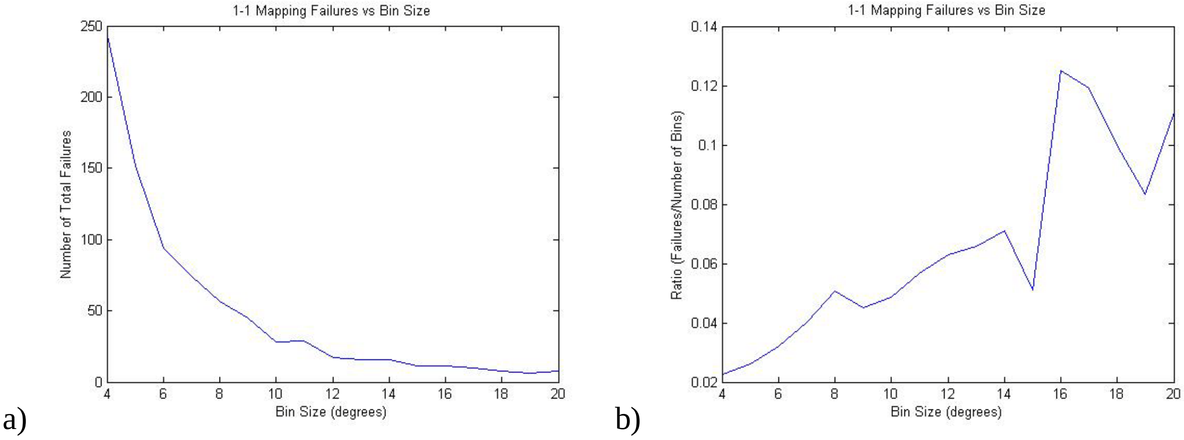


Fig. 15: a) total number of 1-1 mapping failures at each bin size b) ratio of bin failures to total number of bins at each bin size

Another major computational obstacle that is impacted by bin size can be referred to as the dispersion effect. This effect is illustrated in Figs. 16 and 17 where it is clear that the points travel much further than 6 degrees when they are binned between each step. This Dispersion effect is smaller when the rotational steps are the same size as the bins, but a fair amount of dispersion can still take place on the periphery of the bins as the bin boundaries spill over into neighboring bins.

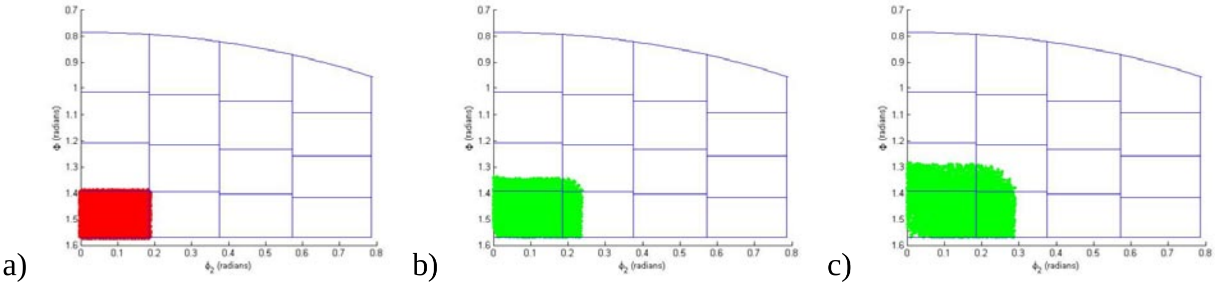


Fig. 16: a)Points in their original bin b)points in 'a' after they have been rotated by 3° around the x-axis without being re-binned c) points from 'b' after they have rotated by an additional 3° (total of 6°) around the x-axis without being re-binned

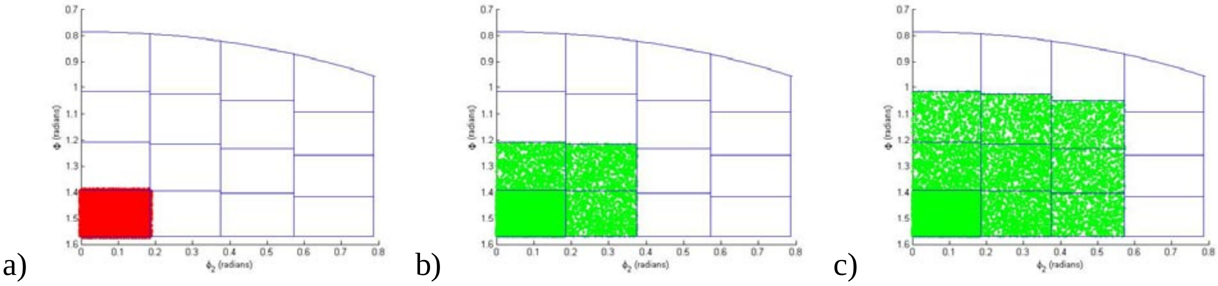


Fig. 17: a)Points in their original bin b)points in 'a' after they have been rotated by 3° around the x-axis and re-binned c) points from b after they have rotated by an additional 3° (total of 6°) around the x-axis and then re-binned again

3.3 **Implementation**

In seeking to implement the streamline algorithm, there are three main constraints that should be met

1) Each streamline should form a closed loop. In other words, after a rotation of 360° all streamlines should map back into their original bins

2) Separate Streamlines should not cross, i.e. occupy the same bin. If the streamlines are not a 1-1 map then this could lead to streams entering a bin on one streamline and leaving on another, negating the intent of constraint 1

3) Each rotational step on the streamline should enter a new bin, otherwise the rotation does not register as a rotation at all. This makes the average bin size the lower bound on the rotational step size

If any of these constraints are not met then the repeatability of the rotational process is seriously compromised. Through experimentation we determined that constraint 3 is always met when the rotational step size is equal to or greater than the average bin dimension. Thus, all experiments shown below use the average bin length as the lower bound on the rotational step size. Constraints 1 and 2 are much more difficult to satisfy and frequently conflict with each other. The three methods below attempt to satisfy these conflicts while minimizing the errors from the obstacles discussed above.

3.3.1 Flowlines

The most basic method for creating a streamline is to incrementally rotate each of the bin points 360˚ while recording the position of these points at consistent intervals (as noted above the step size of these intervals is determined by the average bin length). We will refer to these unbinned rotational paths as flowlines (Fig. 10). Then, only after the full rotation has been recorded, each of the intermediate points is ‘snapped’ to the center of the bin it occupies. This process ensures that constraint 1 is always satisfied (the final rotational point is always in the bin where its flowline originated), but once the binning occurs it causes frequent violations of constraint 2 due to the ‘clumping’ discussed above.

In an attempt to increase the accuracy of this process we attempted to represent the volume of the bin by simultaneously rotating seven points in each bin. Six of these points were centered around the centroid of each bin, with the centroid itself being the seventh point. Each of these points was rotated and binned independently. Then each set of seven streamlines was reduced to one streamline by selecting the mode of the new bins at each step. This process helps to decrease the impact of the dispersion effect mentioned above, but it still fails to eliminate the 'clumping' and satisfy constraint 2.

3.3.2 One-Step: No Constraints

The second method we attempted was to manually rotate each of the bins only one rotational step (from the parent bin to the nearest neighbor in the rotational direction). We then extrapolated this first step to achieve a full 360˚ rotation. In calculating this first step we once again used the seven point method to represent the volume of the bin. This process continues to produce the 'clumping' problem discussed above, and thus violates constraint 2 (the data in Fig. 15 comes from the first step of this method). Of potentially greater concern, this method regularly violates constraint 1 (many of the streamlines to not connect back to bin of origin after a rotation of 360˚). This may be due to the error introduced as you 'snap' each rotational step to the bin centroid. As these rotations are summed together, this error is compounded with each step. In addition, even with no rotational error, the violations of constraint 2 means that the streamlines have been crossed at multiple places, meaning that only one of the crossed streamlines can return to its original bin.

3.3.3 One-Step: Force Constraint 2

Another method we attempted used the same basic idea as the one-step method described above, but it forced the satisfaction of constraint 2 as the points are binned following the first step. In this process we only used the centroid points instead of the seven point method. After the first rotational step, the distance from each rotated point to every original bin centroid was calculated. To assign the rotated points to their new bins, a rotated point is chosen at random and assigned to the nearest bin, as long as that bin is not already occupied. If that bin is occupied the point is assigned to the closest unoccupied bin. All of the rotated points are successively chosen at random until all of the rotated points have been assigned to a new bin.

While this process ensures the enforcement of constraint 2, some of the rotated points are assigned to new bins that are more than three full bin lengths away from their rotated position (over 50˚ when the bin length is 15˚). The fact there are cases where ten or more of the closest bins to a rotated point are already occupied is further evidence of 'clumping'. Another flaw in this process is that the distance calculations we used do not take into consideration the periodic nature of either Euler space or the fundamental zone. The final nail in the coffin of this method is that, while it satisfies constraint 2, it does not satisfy constraint 1 (unless of course constraint 1 is arbitrarily enforced, in which case the streamline ceases to retain a connection to reality)

3.4 **Streamline Conclusions**

The usefulness of the streamline method depends on how important absolute rigidity in constraint 2 is. If it is not essential, then the flowline method could be used to produce an extremely fast, constant time process for the rotation of ODFs. However, until the bin size can be significantly reduced it is likely that the error introduced by the crossed streamlines will dramatically offset any benefits in time optimization the method offers.

4 PRINCIPAL ORIENTATIONS

The next method that we will discuss has been implemented with positive results by Fast, et. al. [16] and is referred to as the principal orientation method. This Method operates directly on the Fourier coefficients in the GSH and FFT representations of the ODF and can make these Fourier representations faster and even more efficient. The concept of principal orientations is derived directly from the definition of the microstructure hull as a convex hull [8, 17, 21, 22] This means that all orientation points that are not on the surface of the hull can be written as linear combinations of surface points. Among this set of surface points, there is an even smaller subset of points that lie at the relative vertices on the hull's surface. This subset of points can serve as a practical basis for the microstructure hull and constitutes what we will refer to as the principal orientations.

4.1 **Background**

As is mentioned in Chapter 1, when the full set of possible orientation in Euler Space (Fig. 18) is converted to spectral form, the Fourier coefficients fill a region in Fourier space (Fig. 19). The size and shape of this region is dependent on the crystal symmetry and on the type of Fourier basis functions used to calculate the ODF. For simplicity, all microstructure hulls referenced in this paper will be produced using cubic symmetry and the GSH Fourier basis (Eqs. 5-7).

The Microstructure hull is one of several major pieces of the MSD framework that only require Fourier coefficients to represent the useful information. Thus, when the hull is acted upon in a uniform manner (rotated, etc) the basis functions do not need to be recalculated, unless they are needed to recalculate the new coefficients.

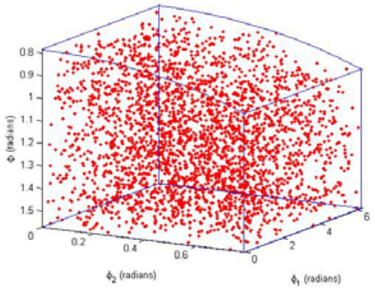


Fig. 18: 3,000 orientation points filling the cubic fundamental zone

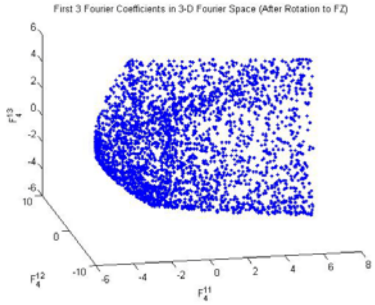


Fig. 19: Orientation points in Fig 17 transferred into Fourier space

However, in the current, brute force, processes the rotations occur at the Euler angle level and therefore the entire ODF, including the *T* (Eq. 6) and *P* (Eq. 7) functions, must be recomputed every time any rotation or other deformation is applied to the material. By operating directly on a small subset of the coefficients that serve as a basis for the convex hull, the principal orientations method avoids unnecessary calculations and significantly increases the speed of the rotations.

It is apparent from Fig. 19 that not all of the points are necessary to provide a complete description of the occupied region. Since this region has already been shown to be convex [8, 17, 21] the outermost points (those at the vertices of the surface, Fig. 20a) can be connected by n-dimensional hyperplanes to completely encapsulate and therefore define the hull region (Fig. 20b).

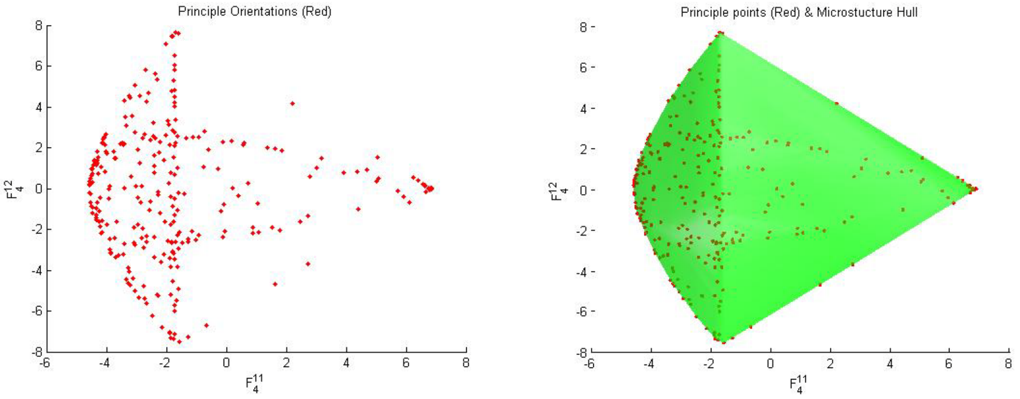


Fig. 20: a) The principal orientation points at the vertices of the region shown in Fig. 19, b) The points shown in part 'a' connected by hyperplanes to form a convex hull (produced using Delaunay Method)

The principal orientations can also be used to recover the interior points using Eq. 39 (notice the similarity to Eq. 11). In this equation, the set of all principal orientations is denoted by M^p, and the superscript *p* is used to signify that this set is based on principal orientations.

$$\tilde{M} = \left\{ F_l^{mn} \middle| F_l^{mn} = \sum_p \alpha_p \, {}^pF_l^{mn},\ {}^pF_l^{mn} \in M^p, \alpha_p \geq 0, \sum_p \alpha_p = 1 \right\} \tag{39}$$

4.2 Selection of Principal Orientations

Once again, it is important to remember that Fourier space usually has more than three dimensions. It is this higher dimensionality that also makes it more difficult to determine which orientations are at the vertices / regions of high surface curvature and which are interior to the hull. We have primarily used two methods for the selection of the principal orientations. The Delaunay triangulation method is used when we are dealing with lower numbers of dimensions (less than 8 dimensions) and the Fukuda method is used when we are dealing higher numbers of dimensions.

4.2.1 Delaunay Selection Method

In some situations only a few Fourier dimensions are needed in the calculations. For example, when working in the elastic region only the first four Fourier dimensions are used. In these situations the Delaunay Triangulation method is a very fast and easy way to gain the principal orientations. The Delaunay algorithm has been implemented in the Convhull function in Matlab. However, this algorithm can only handle data sets with less than 8 dimensions. In the plastic region of microstructure deformation there are 24-80, or more, significant dimensions.

4.2.2 Fukuda Selection Method

The algorithm that we use to find principal orientations for larger (more than nine) Fourier dimensions was suggested by Komei Fukuda in [24]. This algorithm is implemented by initially choosing a large number of random points in the hull. Fig. 21 shows that the actual quantity of these original points does not have a significant impact on the number of principal orientations that are selected. Therefore we chose to use 3,000 original points in most of the calculations shown below. Each point in n-dimensional Fourier space is a 1xn vector. Starting with the first point in the set, each of these points is iteratively considered to determine if it is redundant (can be described using a linear combination of points already in the “hull vertex” set) or non-redundant (outside the span of the hull defined by the current set of vertices). When a point is non-redundant it means that it is needed to create the hull (it is outside the current hull) so it is added to the hull definition set. If the point is determined to be non-redundant it is then added to the set of vertices and the next point is analyzed.

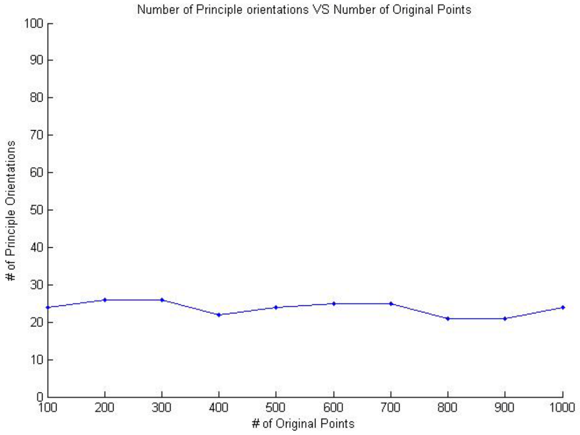


Fig. 21: A graph showing the number of principal orientations produced for a chosen number of 'original' random points used to fill the hull in Fourier space (produced using 6 Fourier dimensions and a tolerance of 0.24)

To determine the redundancy of each point we used a linear programming (LP) technique (as implemented in the linprog function in Matlab). The purpose of a linear programming function is to find a maximizer or minimizer of a linear function subject to linear inequality constraints [24]. The general form of a LP is shown in Eq. 40 (*lb* is the upper bound and *ub* is the lower bound for *x*).

$$\min_x f(x) := c^T x \quad \text{such that} \begin{cases} A \cdot x \le b \\ lb \le x \le ub \end{cases} \tag{40}$$

To solve our specific problem we will first set up a linear feasibility function that has no objective function (Eq. 41). In other words, we are seeking do determine if the point *q* is in the convex hull *S* defined by $S = \{p_1, p_2, \ldots, p_n\}$.

$$\begin{aligned} &\text{Find} && q \\ &\text{Satisfying} && q = \sum_{i=1}^{n} \lambda_i p_i \\ & && \sum_{i=1}^{n} \lambda_i = 1 \\ & && \lambda_i \ge 0 \text{ for all } i = 1, \ldots, n \end{aligned} \tag{41}$$

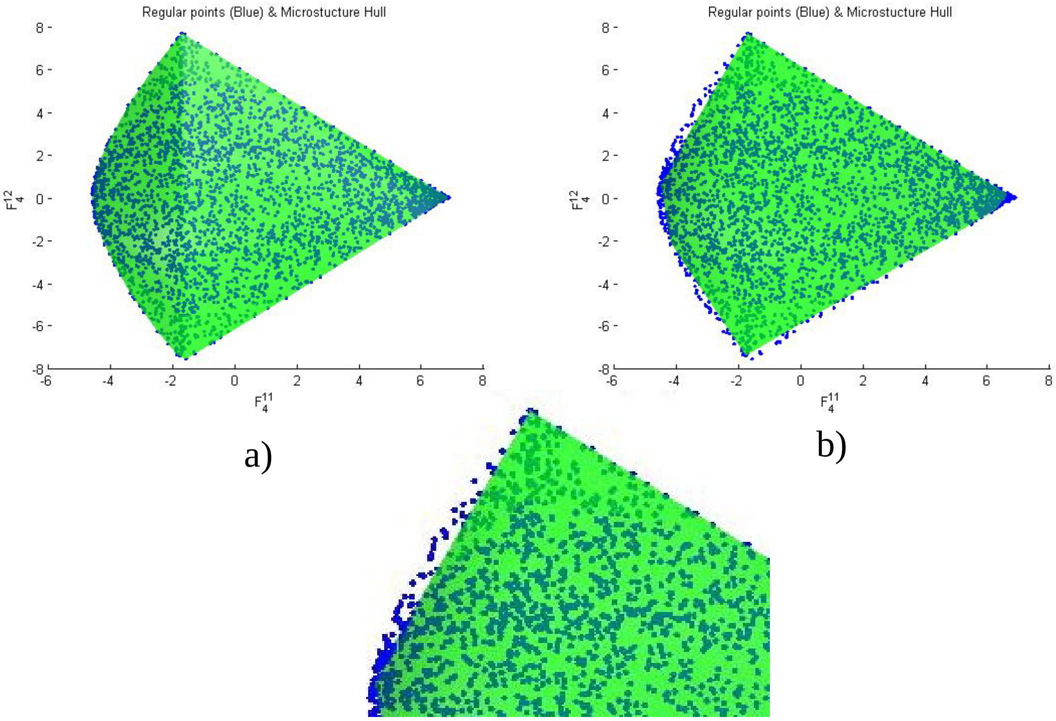
The linear feasibility problem in Eq. 41 has a solution if and only if there is no solution to Eq. 42.

$$\begin{array}{ll} \text{Find} & z_0 \in R \text{ and } z \in R^d \\ \text{Satisfying} & z^T p_i \le z_0 \text{ for all } i = 1,\dots,n \\ & z^T q > z_0 \end{array} \qquad (42)$$

If there exists a solution (z_0, z) to Eq. 42 then there is a hyperplane in $\boldsymbol{R}^d$, defined by the set $H = \{x \in R^d : z^T x = z_0\}$, that separates the polytope $conv(S)$ from the inquiry point *q*, thus making *q* a non-redundant point [24]. In order to solve Eq. 42 we can define Eq. 43, from which we can deduce that *q* is non-redundant if and only if the optimal value f^* in Eq. 43 is strictly positive.

$$f^* = \begin{array}{ll} \text{Maximize} & z^T q - z_0 \\ \text{Subject to} & z^T p_i - z_0 \le 0 \quad \text{for all } i = 1,\dots,n \\ & z^T q - z_0 \le 1 \end{array} \qquad (43)$$

Since the linprog function in Matlab minimizes the result rather than maximizing it, any point that returns a negative result is a non-redundant point and can be added to the definition of *S* as a principal orientation. For computational reasons we have chosen to allow some leeway in this result. Therefore, we have established a tolerance that will allow some points that return a small negative result to be treated as redundant rather than non-redundant. Allowing this tolerance significantly decreases the number of principal orientations that are chosen and this in turn significantly decreases the computation time for the rotations. But, as is seen from Fig. 22, it also shrinks the hull borders and results in some of the original random points being left outside the hull.

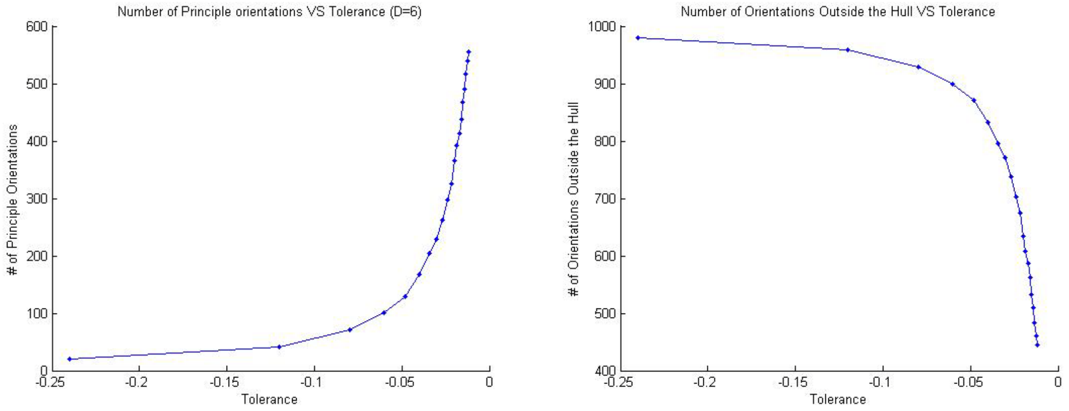


a)

b)

c)

Fig. 22: a) A Microstructure hull calculated using the Delaunay Method, b) A Microstructure Hull calculated using the Fukuda method (using 1,000 random points and a tolerance of 0.06), c) a zoomed in view of the hull in 'b'

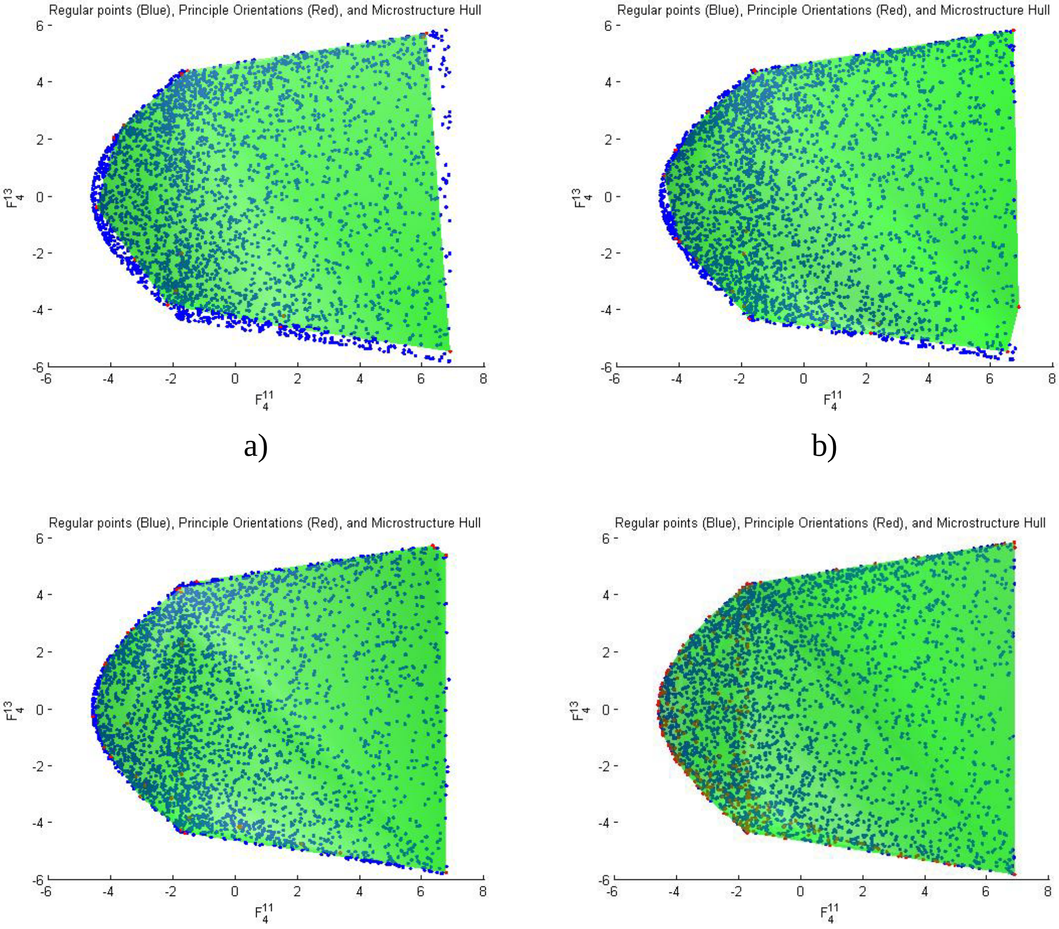


a)

b)

Fig. 23: a) The number of principal orientations used by the Fukuda method to describe a microstructure hull compared to the size of tolerance used, b) The number of original orientation points left outside the hull by the Fukuda method compared to the tolerance used

Fig. 23 shows how the resulting number of principal orientations and the total number of orientations left outside the hull are related to the tolerance that is chosen (the tolerance is shown as a negative number). These graphs also show the tradeoff between the accuracy of the hull and the number of principal orientations needed. Fig. 24 gives a visual representation of the impact that the tolerance has on the relationship between the hull and its perimeter points.



a)

b)

c)

d)

Fig. 24: Microstructure hulls produced using 3,000 random orientation points, 3 Fourier dimensions, and a tolerance of 0.12 (a), 0.06 (b), 0.03 (c), and 0.0 (d)

Since the process used to determine which points are included in the set of principal orientations automatically includes the first several points tested, whether they are vertices or not, we run the algorithmic process twice. The second time we remove each point, p_i, from the hull definition set, *S*, and run the linear programming method described above to test whether the removed point is really non-redundant. If it is redundant it is discarded, but if it is non-redundant it is placed back in *S* and the algorithm considers the next point

4.3 Recovery

Once the principal orientations have been obtained in Fourier space the relationship between these orientations and the rest of the Fourier orientations can be calculated through the linear combination found in Eq. 39. Since F_l^{mn} and $^pF_l^{mn}$ are already known it is only necessary to find the α_p terms to complete the relationship. This is easily accomplished using the '/' operator in Matlab such that $\alpha_p = F_l^{mn} / {}^pF_l^{mn}$. After this relationship has been fully defined, the principal orientations can be rotated independently of the F_l^{mn} terms. As long as the rotation acts uniformly across all principal orientations then the α_p coefficients remain valid and the new, rotated F_l^{mn} terms can be recovered at any time using Eq. 39.

4.4 Principal Orientation Results

To test the validity of using the principal orientations method to rotate ODFs we rotated a set of 3,000 orientations that were originally located near the center of the fundamental zone. Fig. 25-a shows the original state of this texture in both Euler and pole figure form while Fig. 26 represents this ODF using the first three Fourier coefficients for each orientation. For a control or reference rotation, we first rotated this texture in 10° steps for a total of 30° using the brute force method (Fig. 25-b through 25-d).

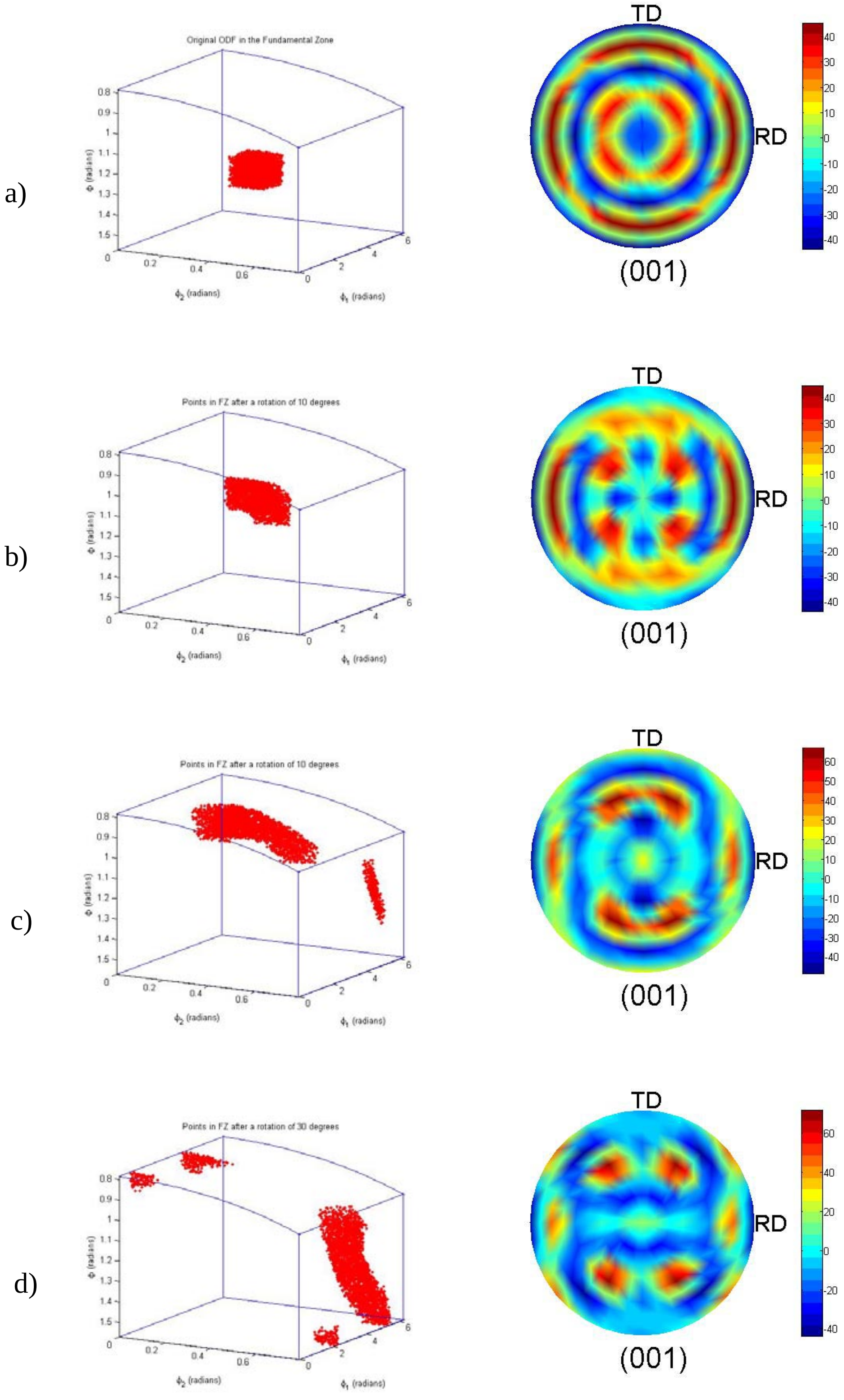


Fig. 25: Euler and pole figure representations of an ODF rotated around the x-axis in 10° steps using the brute force algorithm. a) The ODF before any rotations, b) The ODF after a rotation of 10°, b) The ODF after a rotation of 20°, b) The ODF after a rotation of 30°

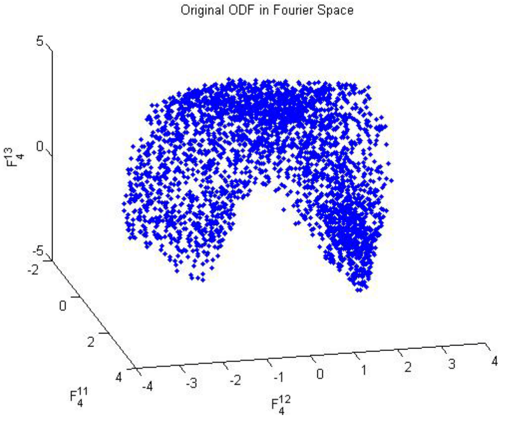


Fig. 26: Fourier space representation of the single crystal points shown in Fig. 25-a

To demonstrate the principle orientations rotation method, we filled the fundamental zone with 3,000 random points and then used the Fukuda method, with 23 Fourier coefficient dimensions and a tolerance of 0.92, to choose 93 principal orientations. We then rotated the 93 Bunge-Euler angles associated with these principal orientations in 10° steps, recovering the Fourier coefficients and plotting the relevant pole figures after each step (right side of Fig. 27). From these pole figures it is apparent that rotational results from the principal orientation method is produces results that are similar to the brute force results, but that there is still a significant amount of error. Table 4-1 attempts to quantify this error by averaging the distances in n-dimensional Fourier space between the secondary and tertiary rotational positions of each point after they have been rotated by the brute force and principal orientation methods. Fig. 28 visually demonstrates this concept by showing the separate rotational paths that result from the two rotation methods. Note that while one plot appears to be much more inaccurate than the other, this is only due to the graphs being 3-dimensional projections of the full 12-dimensional Fourier space. By referring to the titles of the two graphs it is clear that the point in Fig. 28-b actually experiences a smaller rotational error than the point in Fig. 28-a.

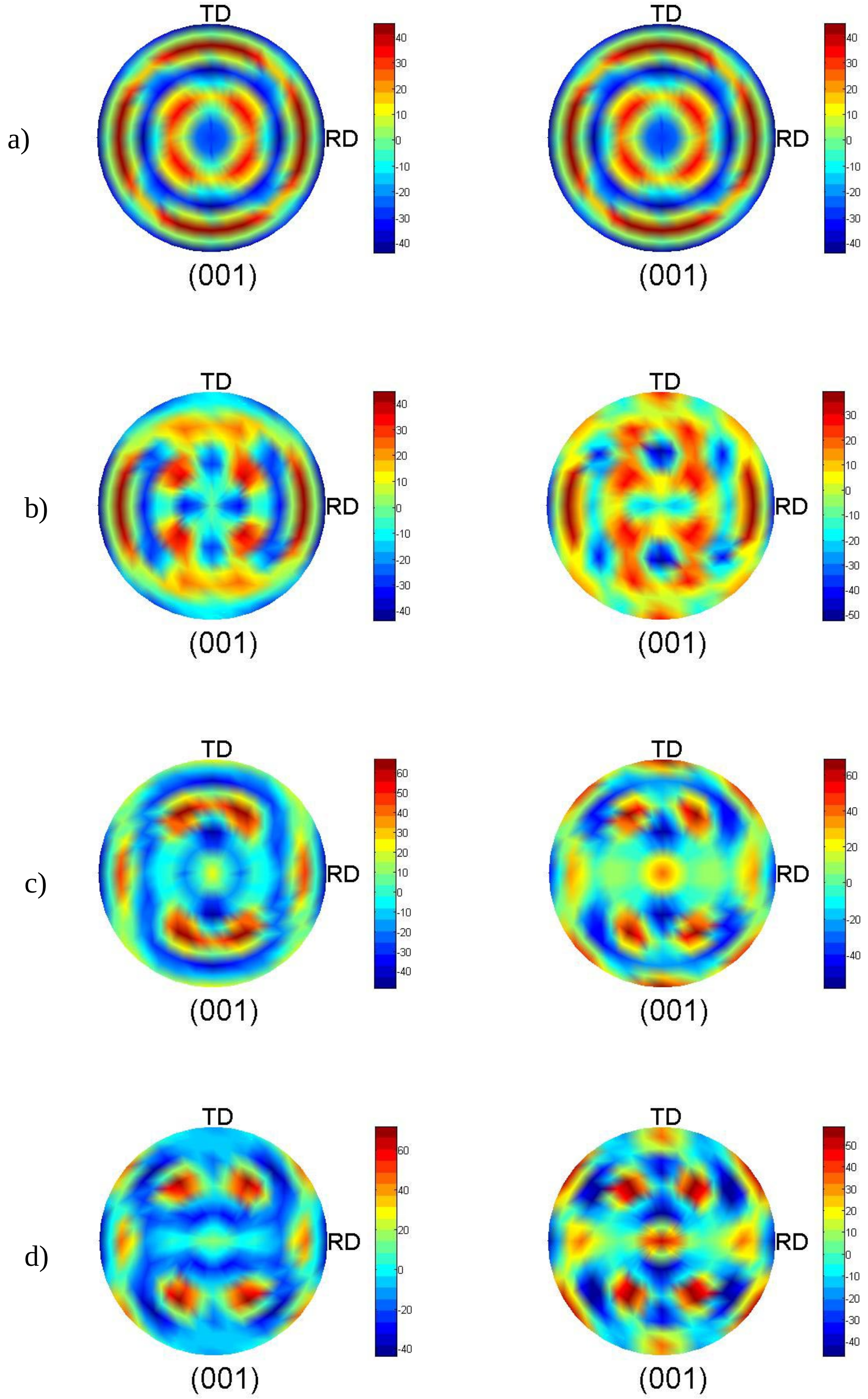


Fig. 27: Comparison of the impact of the brute force (left) and principal orientation (right) rotation methods on a random ODF (created using 3,000 orientation points). a) Before rotation, b) After a rotation of 10˚, c) After a rotation of 20˚, d) After a rotation of 30˚

Table 4-1: Principal Orientation Rotations

			Dimensions			
			8	12	16	23
Tolerance	0.03	# Principal Orientations	787	1727	2245	2780
		Time (sec)	2.68	6.46	7.89	10.9
		Error (10° Rotation)	4.68	7.38	12.13	18.53
		Error (20° Rotation)	6.92	10.41	15.44	20.14
		Error (30° Rotation)	7.65	10.14	14.92	21.81
	0.12	# Principal Orientations	102	313	526	1154
		Time (sec)	0.41	1.11	1.75	4.19
		Error (10° Rotation)	6.26	8.44	9.38	15.43
		Error (20° Rotation)	9.32	12.1	11.5	17.94
		Error (30° Rotation)	9.86	11.77	11.25	17.02
	0.32	# Principal Orientations	41	91	145	320
		Time (sec)	0.17	0.34	0.49	1.14
		Error (10° Rotation)	5.21	8.29	12.55	18.57
		Error (20° Rotation)	8.58	10.91	15.93	20.65
		Error (30° Rotation)	10.96	10.03	14.83	22.57

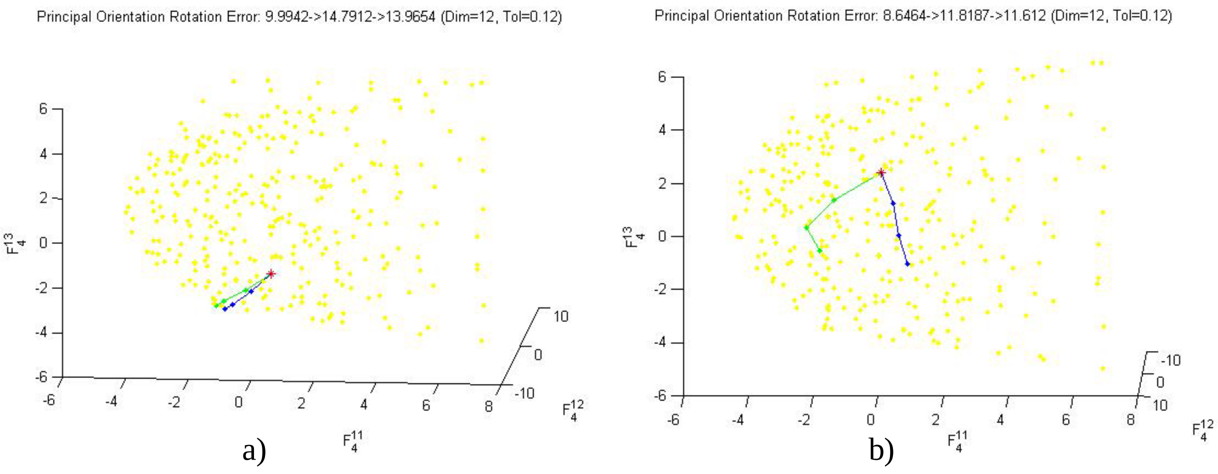


Fig. 28: 3-dimensional representation of the rotation paths followed by two separate points in Fourier space (*-red) as they are rotated using the brute force method (green) and the principal orientation method (blue). The 12-dimension distance measurements for each 10° step are shown in the title of each plot and the yellow points are the un-rotated principal orientations. While it appears that the point in 'b' experiences a larger error, the distance measurements in 12-dimensional space state that 'a' actually experiences a larger error

All of the calculations in Table 4-1 used 3,000 random points in the fundamental zone to calculate the principal orientations and they also all rotated an ODF composed of 3,000 crystals near the center of the fundamental zone. Table 4-1 shows that there is a significant relationship between the number of principal orientations and the time it takes to calculate the rotations. For all of the rotations in Table 4-1 the brute force method took approximately 12.16 seconds. This is almost two seconds slower than the slowest of the principal orientation rotations. Table 4-2 further illustrates how the computational savings provided by the principal orientation method grows as the number of ODF points increases.

Table 4-2: Comparison of Rotation Time (Brute Force vs. Principal Orientations)

Points	Time (Brute Force)	Time (Principal Orientations)
3,000	11.62 sec	0.35 sec
30,000	313.28 sec	0.43 sec

From Table 4-1 we can also learn that the number of principal orientations and the average error increases as the number of dimensions increase. While this result is expected, it is interesting to note that the error values do not seem to significantly improve with an increase in the number of principal orientations or from a tightening of the tolerance.

4.5 Principal Orientation Conclusions

From the results above it appears that while the principal orientations method is much faster than the brute force method it is not very accurate for rotation singles crystals. Essentially, for this method to be accurate the relationship in Eq. 44 must hold true (i.e. any rotation operating on a single crystal in Euler space should maintain the linear relationship between points in Fourier space).

$$R(g) -> R(F) = \alpha_i R(F_i) \tag{44}$$

The evidence above suggests that in our computational application of Eq. 44 $R(g)$ for single crystals does not directly correspond to $\alpha_i R(F_i)$. However, while this method is not very accurate for single crystals it should increase in accuracy as the original ODF approaches a random ODF. In addition, the principal orientations can still be used as an excellent basis for searching the hull as part of the microstructure design framework as shown by Fast, et. al. [16]

5 REFERENCES

1. Adams, B.L. and D.T. Fullwood, *Accessing the Texture Hull and Properties Closure by Rotation and Lamination: Results in the Primitive Basis of Dilation Functions*, in *ICOTOM 2008*. 2008: Pittsburgh.
2. Gelfand, I., R. Minlos, and Z. Shapiro, *Representations of the rotation and Lorentz groups and their applications*. 1963, Oxford: Pergamon Press.
3. Fullwood, D., et al., *Microstructure Sensitive Design for Performance Optimization.* Progress in Materials Science, In Press (available online).
4. Adams, B.L., S.R. Kalidindi, and D.T. Fullwood, *Microstructure Sensitive Design for Performance Optimization*. 2006, Provo, UT: BYU Academic Publishing.
5. Fullwood, D., et al., *Microstructure Sensitive Design for Performance Optimization.* Journal for Progress in Materials Science, In Review.
6. Duvvuru, H.K., Knezevic.M., Mishra,R. K., Kalidindi, S.R., *Application of Microstructure Sensitive Design to FCC Polycrystals.* Materials Science Forum, 546-549, 2007., 2007(546-549): p. 675-680.
7. Houskamp, J.R., *Microstructure Sensitive Design: A Tool for Exploiting Material Anisotropy in Mechanical Design*. 2005, Drexel University, PhD Thesis: Philadelphia. p. 125.
8. Adams, B.L., et al., *Microstructure-sensitive design of a compliant beam.* Journal of the Mechanics and Physics of Solids, 2001. **49**(8): p. 1639-1663.
9. Adams, B.L., S.I. Wright, and K. Kunze, *Orientation imaging: the emergence of a new microscopy.* Metallurgical Transactions A (Physical Metallurgy and Materials Science), 1993. **24A**(4): p. 819-31.
10. Adams, B.L., *Orientation imaging microscopy: Emerging and future applications.* Ultramicroscopy: Proceedings of the 1996 6th Conference on Frontiers in Electron Microscopy in Materials Science, Jun 4-7 1996, 1997. **67**(1-4): p. 11-17.
11. Bunge, H.-J., *Texture analysis in materials science. Mathematical Methods*. 1993, Göttingen: Cuvillier Verlag.
12. Pospeich, J., A. Gnatek, and K. Fichtner, *Symmetry in the Space of Euler Angles.* Kristall und Technik, 1974. **9**(7): p. 729-742.
13. Bunge, H.J. and C. Esling, eds. *Quantitative Texture Analysis*. 1981, Societe Francaise De Metallurgie.
14. Binci, M., D. Fullwood, and S.R. Kalidindi, *A new spectral framework for establishing localization relationships for elastic behavior of composites and their calibration to finite-element models.* Acta Materialia, 2008. **56**(10): p. 2272-2282.
15. Duvvuru, h.k., *Spectral Methods for Modeling Microstructure Evolution in Deformation Processing of Cubic Polycrystalline Metals*, in *Mechanical Engineering*. 2007, Drexel University.

16. Tony Fast, M.K., Surya R. Kalidindi, *Application of Microstructure Sensitive Design to Structural Components Produced from Hexagonal Polycrystalline Metals.* Computational Materials Science, 2008. **43**: p. 374-383.
17. Schwartz, A.J., M. Kumar, and D.P. Field, *Electron Backscatter Diffraction in Materials Science*. 2nd ed, ed. B.L. Adams. 2009, New York: Springer. 432.
18. Bunge, H., *Texture Analysis in Materials Science.* Butterworths, 1982.
19. Kalidindi, S.R., et al., *Microstructure sensitive design of an orthotropic plate subjected to tensile load.* International Journal of Plasticity, 2004. **20**(8-9): p. 1561-1575.
20. Adams, B.L., M. Lyon, and B. Henrie, *Microstructures by design: linear problems in elastic-plastic design.* International Journal of Plasticity, 2004. **20**(8-9): p. 1577-1602.
21. Kalidindi, S.R., et al., *Elastic properties closures using second-order homogenization theories: Case studies in composites of two isotropic constituents.* Acta Materialia, 2006. **54**(11): p. 3117-3126.
22. Rockafellar, R.T., *Convex Analysis.* Princeton University Press, 1970: p. 19.
23. Morawiec, A., *Orientations and rotations - computations in crystallographic textures*. 2004: Springer.
24. http://www.ifor.math.ethz.ch/~fukuda/polyfaq/polyfaq.html. *Frequently Asked Questions in Polyhedral Computation*. 2004 [cited 2009 9 Dec].