Improved Prediction of Glass Fiber Orientation in Basic Injection Molding Geometries

By

Kevin J. Meyer

Dissertation submitted to the faculty of

Virginia Polytechnic Institute and State University

In partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

In

Chemical Engineering

Donald G. Baird, Chairman

Luke E. Achenie

Stephen M. Martin

Peter Wapperom

September 2nd, 2013

Blacksburg, VA

Keywords: Injection Molding, Composites, Glass Fiber, Numerical Modeling

Improved Prediction of Glass Fiber Orientation in Basic Injection Molding Geometries

Kevin J. Meyer

Abstract

This work is concerned with the prediction of short (SGF) and long glass fiber (LGF) orientation in a center-gated disk and end-gated plaque injection molding test geometry using a simulation method that has not been attempted previously. Previous work has used assumptions to simplify the fiber orientation geometry (assuming a thin cavity) or flow field (neglecting fountain flow and entry regions). LGF orientation is predicted in a center-gated disk injection molding geometry including the advancing front and simulating the sprue and gate region (SGM method) so that no assumption about fiber orientation at the mold entrance has to be made. Using a semi-flexible fiber model and orientation parameters obtained through rheology, increased agreement was found between predicted and experimentally obtained values of orientation using the SGM method and a semi-flexible fiber model than was found using a Hele-Shaw approximation. The SGM method was applied to the end-gated plaque to predict SGF orientation both along and away from the centerline using an objective (reduced strain closure model) and non-objective (strain reduction factor model) orientation model. The predicted values of the strain reduction factor model showed reasonable agreement with experimentally obtained values of orientation throughout the three-dimensional cavity when using orientation parameters fit to experimental orientation data. Furthermore it was found that the objective model predicted results very similar to the non-objective model suggesting that objectivity may not play a role in predicting orientation in more complex geometries such as an end-gated plaque. Finally, the SGM method was applied to the end-gated plaque geometry to predict LGF orientation using a rigid and semi-flexible fiber model. It was found that the SGM method and the semi-flexible fiber model provides orientation predictions that are similar to experimentally obtained values of orientation.

Original Contributions

- Improved Long Glass Fiber Orientation Predictions by Combining a Semi-Flexible Fiber with Gate Effects and the Advancing Front

- Utilized a Simulation Technique to Predict Short and Long Glass Fiber Orientation at a Number of Points in Complex Molding Geometries (along and away from centerline of mold)

- Developed a Method for Determining Orientation Parameters for Use in Simulating Short and Long Glass Fiber Orientation in an End-Gated Plaque

Acknowledgements

The author wishes to express his thanks and gratitude to Dr. Donald G. Baird for his guidance and support that resulted in the completion of this work. Additionally, the author would like to thank his committee members Dr. Luke Achenie, Dr. Stephen Martin and Dr. Peter Wapperom for their comments and suggestions during the evolution of this work.

The author would also like to acknowledge the following people:

- Ms. Amy Poli for her support and belief in me and for her unending patience. Without her support this work would not have been completed.
- Mrs. Laura Brown and Ms. Sandra Meyer for their support and love through this difficult project and helping mold me into the man that I am today.
- Mr. Edward Richards and Mrs. Marlene Richards for helping me with life's problems, big and small, and making sense of the world.
- Ms. Julie Love for her support and friendship through this arduous process.
- Ms. Kimberly Agnello for helping me edit this document and providing me with constructive criticism.
- To the close friends that provided support during this project (in no particular order): Ashley Gordon, Adam Larkin, Kevin Norfleet, Andy Ford, Hillary Paulson, Tim Vadala, Caitlin Cossaboom, Lucas Vu, Colin Spellmeyer, Matt Sobieski, Ryan Cardillo, Andrew Lane and Shawn Hamm.
- The past and current members of the Polymer Processing Lab at Virginia Tech (in no particular order): Mike Heinzer, Kevin Ortman, Syed Mazahir, Chen Chen, Gregorio Vélez-García, Chris McGrady, Kevin Harrington, Mark Cieslinski, Chen Qian and Hongyu Chen.
- The staff of the chemical engineering department at Virginia Tech whose help was irreplaceable: Mrs. Diane Cannaday, Mrs. Tina Russell, Mrs. Nora Bentley, Mrs. Jane Price, Mr. Riley Chan and Mr. Mike Vaught.

Format of Dissertation

This dissertation is written in journal format. Chapters 3, 4 and 5 are self-contained papers that are to be submitted for journal publication. Each paper separately described the experiments, results and conclusions relative to each chapter. With the exception of Chapter 2 the figures and tables are presented after the reference section of each chapter.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1 Introduction

A polymer composite typically consists of a resin and some type of fiber additive. Glass fibers are often a popular choice as an additive and have been readily used since the 1950's [1]. A glass fiber additive acts as a reinforcing phase and can provide increases in stiffness and strength of a material [2, 3]. Long glass fibers $(L > 1 \text{ mm})$ are of particular interest because of the advantageous properties they provide over short glass fibers $(L < 1 \text{ mm})$ in injection molded parts.

It has been shown and observed that the orientation of fibers within a sample play a crucial role in the properties of that part [4]. To predict and optimize the properties of the final product, it is highly desirable to be able to accurately model how the fibers orient during the filling of a mold to produce a part. It is for this reason that the study of fiber orientation during processing has been an area of significant research efforts [5].

The injection molding of glass fiber composite systems is a very complex problem usually involving a number of simultaneously occurring phenomena. First, many molds of industrial interest are geometrically complicated and will involve a number of gates and complicated flow regions [6]. Second, the molten composite is almost always a different temperature than the mold which it is being injected into, and the viscosity of the fluid is temperature dependent so that the momentum and energy equations are coupled. Additionally, the fibers within a molten composite also orient in the fluid as a result of the velocity gradients and temperature gradients that arise from being injected into the mold [7]. Complicating the problem further, the fluid filling a mold is a moving boundary problem so the domain of the solution must be continually updated during the solution process. Ultimately, the system to be modeled must be simplified in a number of ways so that the problem becomes tangible to solve in a realistic time frame.

1.1 Solution of Equations for Fiber Orientation

The prediction of long glass fiber orientation in any geometry is a complicated task involving the solution of a number of non-linear equations. The solution to any flow problem begins with the equations of continuity and the equation of motion. A common assumption is that the polymer fluid filling the cavity is in the laminar flow regime which produces continuity and motion equations in the form of Eq. (2.1) and Eq. (2.2):

$$0 = \nabla \cdot \bar{v} \tag{2.1}$$

$$0 = -\nabla P + \nabla \cdot \bar{\bar{\tau}} \tag{2.2}$$

In Eq. (2.1) and Eq. (2.2) $\bar{v}$ is the velocity vector, P is the isotropic pressure and $\bar{\bar{\tau}}$ is the extra stress tensor. Additionally, many simulations involving the flow of a polymeric fluid into a cavity include temperature effects so that the heat equation must also be solved. The heat equation is seen in Eq. (2.3) for an incompressible fluid including the effects for viscous dissipation.

$$\rho C_P \frac{DT}{Dt} = k\nabla^2 T + \overline{\overline{\tau}} : \overline{\overline{\dot{\gamma}}} \tag{2.3}$$

In Eq. (2.3) ρ is the density, C_P is the heat capacity, DT/Dt is the material derivative of the temperature, k is the thermal conductivity, $\overline{\overline{\tau}}$ is the extra stress tensor and $\overline{\overline{\dot{\gamma}}}$ is the rate of strain tensor.

The solution of fiber orientation may occur simultaneously with the flow and heat equations ("coupled") or can be solved as a second step in a two-step solution process ("decoupled"). In general, the equations to predict fiber orientation can be a function of a number of factors given by Eq. (2.4):

$$\frac{D\overline{\overline{A}}}{Dt} = f\left(\overline{\overline{A}}, \overline{\overline{\mathbb{A}}}_4, \overline{\overline{D}}, \overline{\overline{W}}, k, l_b\right) \tag{2.4}$$

In Eq. (2.4) $\overline{\overline{A}}$ is the 2nd moment of the orientation distribution (2nd order orientation tensor), $\mathbb{A}_4$ is the 4th moment of the orientation distribution function (4th order orientation tensor), $\overline{\overline{D}}$ is the rate of deformation tensor, $\overline{\overline{W}}$ is the vorticity tensor, k is a fiber's resistivity to bending and l is the length of a fiber. While coupling the flow equations with the orientation equations sometimes yields better results, the complications that arise in the solution of such coupled systems generally outweigh the benefits [4, 8, 9].

1.2 Solution of Fiber Orientation in a Center-Gated Disk

One of the most studied geometries in the prediction of glass fiber orientation is the center-gated disk mold shown in Figure 1.1. The axisymmetric nature of the problem allows the flow calculations to be performed in 1.5 or 2.5 dimensions (Hele-Shaw assumption uses 1.5 dimensions, entire domain uses 2.5 dimensions).

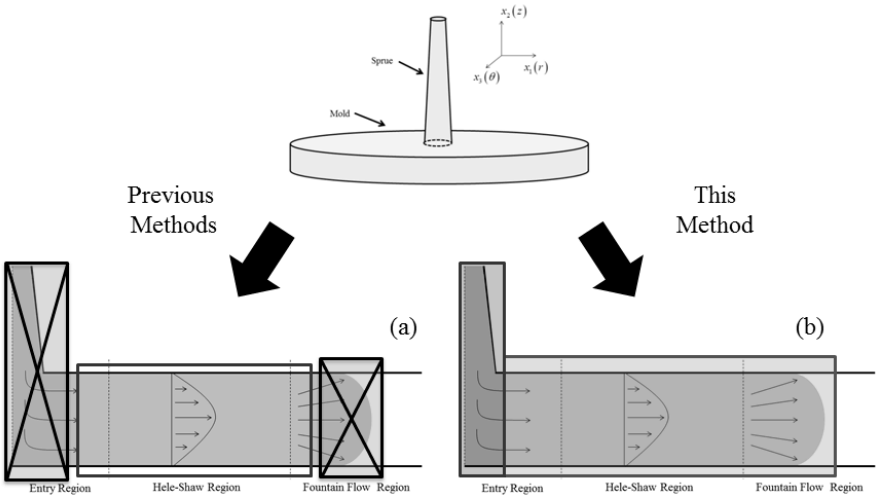

Figure 1.1: Geometry of a center-gated disk. (a) Qualitative representation of work performed up to this point for long-glass fiber systems (Red Square – Hele-Shaw Region, Assumptions are made at gate about fiber region, fountain flow not included). (b) Qualitative

representation of simulation domain for simulating long glass fibers in this work (Blue Square – Simulation includes fountain flow and entry effects, Red Square – Simulation includes sprue).

Within the center-gated disk, simulations are commonly carried out in the "Hele-Shaw" region. This assumes a mainly shear dominating flow field and neglects a frontal flow region, referred to as "fountain flow", and also neglects the entry effects to the mold, referred to as the "gate region". Hele-Shaw flow, represented by Figure 1.1 (a), has been used to simulate flows of fiber-filled systems and is currently utilized in many numerical simulation packages but has been shown to produce less accurate orientation predictions at the walls, near the front, and near the gate of the mold [10, 11]. As the review of literature will show, the modeling of long glass fibers in a center-gated disk has received relatively little attention. One attempt by Nguyen et al. [12] applied a short fiber model to a long fiber system and assumed Hele-Shaw flow. Ortman et al. [13] used the Hele-Shaw assumption in order to simulate the orientation kinetics of LGF's during filling using a semi-flexible fiber model. Instead this work seeks to investigate the accuracy gained in fiber orientation predictions when no assumptions about fiber orientation into the mold cavity are made and the mold is simulated from the sprue/mold interface shown by Figure 1.1 (b).

1.3 Solution of Fiber Orientation in End-Gated Plaque

A more industrially applicable geometry of interest is the end-gated flow domain given by the mold in Figure 1.2. Commonly in the end-gated system, the sprue, runner and gate are neglected and only flow in the mold cavity is simulated [14-17]. The previous work has employed the Hele-Shaw assumption to simplify the three-dimensional geometry into a rectangular cavity, as seen in Figure 1.2 (a).

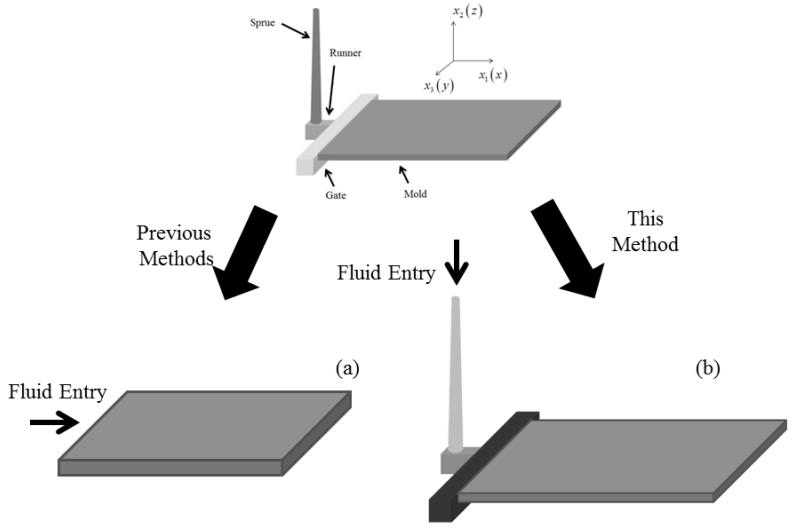

Figure 1.2: Geometry of an end-gated plaque. (a) Solution of fiber orientation through a rectangular cavity using Hele-Shaw assumption (may be combined with fountain flow model) requiring specification of orientation at mold entry. (b) Solution of fiber orientation using the three-dimensional end-gated plaque geometry without assumptions at mold entry.

The modeling of LGF orientation while being injected into a fully three-dimensional mold, such as an end-gated plaque shown above in Figure 1.2 (b), has seen only one effort by Nguyen et al. [12]. In this study a short fiber orientation model was applied to a long glass fiber systems and a Hele-Shaw assumption was made. Unfortunately comparisons between predicted and experimental fiber orientation values were carried out at the center-line of the mold in this study. Hele-Shaw flow makes assumptions about the flow field near the front and near the gate which effect the orientation of fibers. Instead both regions should be included when predicting fiber orientation in the end-gated mold. Furthermore, the entry to the mold cavity is a complex three-dimensional flow field and the effects should be studied to assess effects on fiber orientation in the mold. Understanding the evolution of long glass fiber orientation in fully three-dimensional test geometries is necessary as it is the foundation for predicting fiber orientation in more industrially relevant molding geometries.

1.4 Research Objectives

The primary goal of this research is to access the increase in accuracy of modeling long glass fiber orientation by modeling the entire injection domain as a single entity in two injection molding test geometries. Three objectives have been proposed to accomplish this goal:

1. Quantify the improvement gained in predicting long glass fiber orientation in a center-gated disk by simulating the sprue, gate and mold (SGM method) as a single domain using established orientation equations and rheologically determined orientation parameters and comparing results to experimentally obtained fiber orientation data.

2. Assess the accuracy of the SGM method to predict short glass fiber orientation in an end-gated plaque mold by comparing predicted fiber orientation values (obtained from an objective and non-objective rigid fiber model) to experimentally measured fiber orientation both along and away from the centerline.

3. Assess the accuracy of the SGM method to predict long glass fiber orientation in an end-gated plaque mold by comparing predicted fiber orientation values (obtained from both a rigid and semi-flexible fiber model) to experimentally determined fiber orientation both along and away from the centerline.

1.5 References

[1] C. C. Lee, F. Folgar, and C. L. Tucker, "Simulation of Compression Molding for Fiber-Reinforced Thermosetting Polymers," *Journal of Engineering for Industry-Transactions of the Asme,* vol. 106, pp. 114-125, 1984.

[2] B. T. Åström, *Manufacturing of polymer composites*, 1st ed. London: Chapman & Hall, 1997.

[3] S. G. Advani, *Process modeling in composites manufacturing*. New York: Marcel Dekker, 2003.

[4] B. E. VerWeyst, C. L. Tucker, P. H. Foss, and J. F. O'Gara, "Fiber orientation in 3-D injection molded features - Prediction and experiment," *International Polymer Processing,* vol. 14, pp. 409-420, Dec 1999.

[5] S. G. Advani and E. M. Sozer, *Process modeling in composites manufacturing*. New York: Marcel Dekker, 2003.

[6] A. Whelan, *Injection moulding machines*. London: Elsevier Applied Science, 1984.

[7] A. Larsen, "Injection molding of short fiber reinforced thermoplastics in a center-gated mold," *Polymer Composites,* vol. 21, pp. 51-64, Feb 2000.

[8] B. E. VerWeyst and C. L. Tucker, "Fiber suspensions in complex geometries: Flow/orientation coupling," *Canadian Journal of Chemical Engineering,* vol. 80, pp. 1093-1106, Dec 2002.

[9] G. M. Velez-Garcia, S. M. Mazahir, P. Wapperom, and D. G. Baird, "Simulation of Injection Molding Using a Model with Delayed Fiber Orientation," *International Polymer Processing,* vol. 26, pp. 331-339, Jul 2011.

[10] R. S. Bay and C. L. Tucker, "Fiber Orientation in Simple Injection Moldings .2. Experimental Results," *Polymer Composites,* vol. 13, pp. 332-341, Aug 1992.

[11] E. K. Henry, S.;Kennedy P., "Fiber orientation and the mechanical properties of SFRP parts.," presented at the SPE Tech. Papers, 1994.

[12] B. N. Nguyen, S. K. Bapanapalli, J. D. Holbery, M. T. Smith, V. Kunc, B. J. Frame*, et al.*, "Fiber length and orientation in long-fiber injection-molded thermoplastics - Part I: Modeling of microstructure and elastic properties," *Journal of Composite Materials,* vol. 42, pp. 1003-1029, May 2008.

[13] K. Ortman, D. Baird, P. Wapperom, and A. Aning, "Prediction of fiber orientation in the injection molding of long fiber suspensions," *Polymer Composites,* vol. 33, pp. 1360-1367, 2012.

[14] G. G. Lipscomb, M. M. Denn, D. U. Hur, and D. V. Boger, "The Flow of Fiber Suspensions in Complex Geometries," *Journal of Non-Newtonian Fluid Mechanics,* vol. 26, pp. 297-325, Jan 1988.

[15] C. L. Tucker, "Flow Regimes for Fiber Suspensions in Narrow Gaps," *Journal of Non-Newtonian Fluid Mechanics,* vol. 39, pp. 239-268, May 1991.

[16] M. Altan, S. Subbiah, S. I. Guceri, and R. B. Pipes, "Numerical Prediction of Three-Dimensional Fiber Orientation in Hele-Shaw Flows," *Polymer Engineering and Science,* vol. 30, pp. 848-859, 1990.

[17] K. H. Han and Y. T. Im, "Numerical simulation of three-dimensional fiber orientation in injection molding including fountain flow effect," *Polymer Composites,* vol. 23, pp. 222-238, Apr 2002.

Chapter 2

Literature Review

Preface

This chapter provides a review of the literature relevant to this research. This portion of the document is divided into five major subsections: general classifications of glass fiber composites, determination of experimental fiber orientation in glass fiber composites, modeling equations for the stress tensor used with glass fiber suspensions, modeling equations for the prediction of orientation of glass fibers in suspension and a summary of the prediction of glass fiber orientation in different geometries. The figures and tables are given in the text as needed.

2 Review of Literature

In the following section, a review of literature concerning glass fiber composites is presented. The following text is a review of pertinent information on glass fiber reinforced composites which allow the reader to be fluent enough in the subject matter to understand the concepts presented throughout this thesis. First, classification of fiber systems based on concentration and fiber length is reviewed in § 2.1. Next, in § 2.2 determination of experimental fiber orientation in glass filled thermoplastics is reviewed. Rheologically obtained parameters fit to stress models are used in rigid and semi-flexible fiber orientation simulations. Therefore, modeling of the stress tensor in rigid and semi-flexible fiber suspensions is reviewed in § 2.3. Next, modeling fiber orientation for rigid and semi-flexible fibers in a flow field is discussed in §2.4. Finally, predicting both rigid and semi-flexible fiber orientation during processing is reviewed in § 2.5.

2.1 Glass Fiber Composites

Additives have long been used in polymers as a way to change the properties of a matrix. Fiber additives are used to increase rigidity and enhance mechanical properties. One such example of this is the use of glass fibers in the design of a bus seat reported by Bartus et al. [1]. The findings showed a 43% weight reduction and 18% cost reduction while maintaining the same standards of previous, non-composite alternates. Parts made from glass-fiber reinforced plastics have also been reported to have a high strength to weight ratio along with high stiffness and creep endurance [2]. Glass fibers have the ability to be added to a wide range of polymers while generally increasing properties [3].

Classifying a fiber-filled system usually by either length or concentration often dictates the assumptions to be made and equations to be solved. First, the classification of fiber-filled systems by concentration is addressed in §2.1.1. Next, the use of fiber length when classifying fiber filled systems is reviewed in § 2.1.2.

2.1.1 Classification by Fiber Concentration

When classifying the concentration of fibers within a system, one common method employed is given by Doi and Edwards [4] and is based on concentrated polymer solutions adapted for fiber suspensions. This method is based on fiber concentration and is defined by the relation given in Eq. (2.1):

$$\phi = \frac{\pi n L^3}{4 a_r^4} \tag{2.1}$$

Eq. (2.1) is taken to be over a given suspension of fibers where ϕ is the fiber volume fraction, n is the number of fibers per unit volume and L and a_r are the fiber's length and aspect ratio, respectfully. From the definition, Doi and Edwards [4] described three regions of concentration. These regions are termed by the authors as *dilute* (§ 2.1.1.1), *semi-dilute* (§ 2.1.1.2), and *concentrated* (§ 2.1.1.3) regions. Figure 2.1 shows the relationship between fiber aspect ratio, a_r, fiber volume fraction, ϕ, and the regions that combinations of the two can yield.

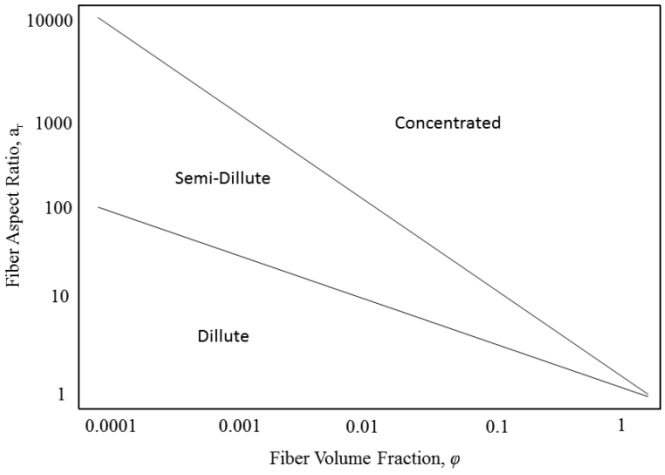

Figure 2.1: Concentration regimes as a function of volume fraction, f , and aspect ratio, a_r . Estimations are done using Dinh and Armstrong theory for aligned fibers [5].

2.1.1.1 Dilute Region of Fiber Concentration

A fiber suspension is defined as dilute when the presence of a single fiber is completely unaffected by any other fibers in the system. This regime exists, theoretically, when the average distance between adjacent fibers is larger than L, the length of a given fiber in the system. Additionally, the number of fibers per unit volume is also restricted to a value of $n < 1/L^3$. The volume fraction is now also confined to a value of $\phi < 1/a_r^2$. A fiber in the dilute system is never able to "observe" another fiber within a system. Specifically, a fiber is unaffected by any other fibers around it either by direct fiber-fiber interaction or hydrodynamic interactions.

2.1.1.2 Semi-Dilute Region of Fiber Concentration

In the semi-dilute region free rotation of a fiber is restricted by the presence of other fibers but fiber collision is still rare. Quantitatively, fiber-fiber interaction can occur when the number of rods per unit volume is over $n > 1/L^3$. The upper limit for a system of completely random rods in a unit volume to be semi-dilute is given by Eq. (2.2) and the upper limit of a system of completely aligned rods is given by Eq. (2.3).

$$\text{Random System} \rightarrow n \ll \frac{1}{dL^2} \tag{2.2}$$

$$\text{Aligned System} \rightarrow n \ll \frac{1}{d^2 L} \tag{2.3}$$

The fiber volume fraction is also restricted to a value of $a_r^{-2} < \phi \ll a_r^{-1}$.

2.1.1.3 Concentrated Region of Fiber Concentration

The concentrated regime encompasses all those systems which are above the limit of the semi-dilute region described in § 2.1.1.2. The quantitative description of this is described as either $n \geq 1/d^2 L$ or $\phi \geq a_r^{-1}$ for the number of fibers per unit volume and fiber volume fraction, respectfully. In this region both hydrodynamic interactions and fiber-fiber interactions are highly prevalent with fiber collisions occurring frequently. In concentrated systems it is also possible to observe behavior similar to liquid-crystalline polymer systems whereby clustering occurs to form a nematic phase of locally preferential alignment [4]. Concentrated systems are inherently complex and relatively little is understood about these systems in comparison to dilute and semi-concentrated regimes.

2.1.2 Classification by Length

The second of the two classifications methods for fiber suspensions is by length of the fibers within the suspension. Generally, systems are classified by either being "short" glass fiber systems (SGF's) or "long" glass fiber systems (LGF's). Short glass fibers have been studied extensively by a number of authors [6-9]. The term "short" is generally used with systems when the fiber length distribution has an average length less than 1mm. Conversely, the term "long" is used when $L > 1$ mm with the distinction based primarily on mechanical properties of the fiber [10]. A LGF system may exhibit the ability to bend, quantified by Switzer and Klingenberg [11, 12] as the effective flexibility of a fiber and given in Eq. (2.4):

$$f^{\text{eff}} = \frac{64\eta_{\text{m}}\dot{\gamma}a_r^4}{E_Y\pi} \tag{2.4}$$

In the equation above, η_{m} is the matrix viscosity, $\dot{\gamma}$ is the shear rate, a_r is the aspect ratio, and E_Y is the Young's modulus of the individual fiber. From the expression in Eq. (2.4), the flexibility of a fiber can be seen to increase with increasing aspect ratio for a given Young's modulus.

Experimentally, fiber suspensions do not have one fixed length but a number of different fiber lengths within a system known as a fiber length distribution (FLD). After the FLD has been measured, the average fiber length is commonly calculated by taking the first and second moments of the fiber length distribution to form the number average length (L_n) and weight average length (L_w) shown in Eq. (2.6) and Eq. (2.5), respectfully:

$$L_n = \frac{\sum n_i L_i^2}{\sum n_i L_i} \tag{2.5}$$

$$L_w = \frac{\sum n_i L_i}{\sum n_i} \tag{2.6}$$

2.2 Determination of Experimental Fiber Orientation

Quantifying experimental fiber orientation is of upmost importance so that theoretical models can be developed with reliable experimental fiber orientation data. Fiber orientation can vary across the length, width, and thickness of a sample so accurately determining experimental fiber orientation is usually difficult and a number of techniques have been investigated including microradiography [13], scanning acoustic microscopy [14], scanning electron microscopy [15], nuclear magnetic resonance imaging [16], x-ray tomographic imaging [17], and reflection microscopy [18]. While many of the above methods show promise, they are limited by the ability to quantify large sample areas or have too poor resolution or contrast. The most promising method is optical microscopy whereby two-dimensional images are taken of the sample and projections are made into the plane to yield a quasi-3-dimensional volume by which to calculate orientation [19, 20]. The sample preparation for optical microscopy is labor intensive, but necessary, to obtain samples of high quality. Once high quality samples are obtained, image acquisition and processing are performed to assist in the elliptical footprint recognition. Once the processing has been completed the method of ellipses (MoE) is used to calculated experimental fiber orientation of the samples in question. Each step of the process will be addressed in separate sections. First, § 2.2.1 covers the composite material sample preparation. Next, § 2.2.2 discusses the image acquisition and stitching procedure. Finally, § 2.2.3 discusses the calculation of fiber orientation based on the method of ellipses.

2.2.1 Sample Preparation for Optical Microscopy

Fiber containing composites must be prepared before optical microscopy can be performed. Sample preparation directly affects the quality of micrographs obtained in the imaging step so care must be maintained throughout the process. The sample preparation steps include cutting, marking, mounting, grinding, polishing, etching, and sputtering. The procedure presented has been tested on a number of glass containing composites including 30 weight percent short fiber polybutlyene terephthalate (SF-PBT, Valox 420, SABIC), 30 weight percent short glass fiber polypropylene (SF-PP, RTP-105, RTP Company), and 30 weight percent long glass fiber polypropylene (LF-PP, LNP Verton MV006S, SABIC).

2.2.1.1 Sample Cutting

Sample cutting is usually a balance between the speed at which a cut can be made and the resulting friction that can occur. Aggressive cutting can fracture the surface of the sample in question and may even cause irregularities in the fibers being analyzed. In order for fibers to be recognized with MoE clear elliptical footprints must be generated. Therefore, the sample is cut using a low-speed diamond wheel saw (South Bay Technology, Inc.) using cutting oil (Precision Surfaces International, Inc.) to reduce the friction imposed on the sample.

2.2.1.2 Sample Marking

In the polishing steps that will be outlined, a very flat surface is required to make accurate predictions about fiber orientation. For this reason the samples are marked once they are cut so the preparer knows how much material to polish off of the mounted sample. Tridimensional sample markers made of a polymeric material are used to identify areas of

interest on the sample. The markers are made of red-colored polyethylene terephthalate and bonded with cyanoacrylate adhesive (Krazy Glue®).

2.2.1.3 Sample Mounting

Once the sample has been cut and marked, it is mounted in a thermoplastic epoxy in the shape of a puck roughly 30 mm in diameter. This is done by placing the samples of interest in 20 ml of acrylic powder (thermoplastic powder, Precision Surfaces International, Inc.), heating to a temperature of 458 K and steadily applying 1 kPa of pressure in a specialized molding apparatus. After heating the sample for 15 minutes, the heating element is removed while still under pressure. The sample is removed only when the thermoplastic powder has solidified around the samples. To keep the samples upright during the puck molding, soft clips (spring clips, Precision Surfaces International, Inc.) are used around the samples in question.

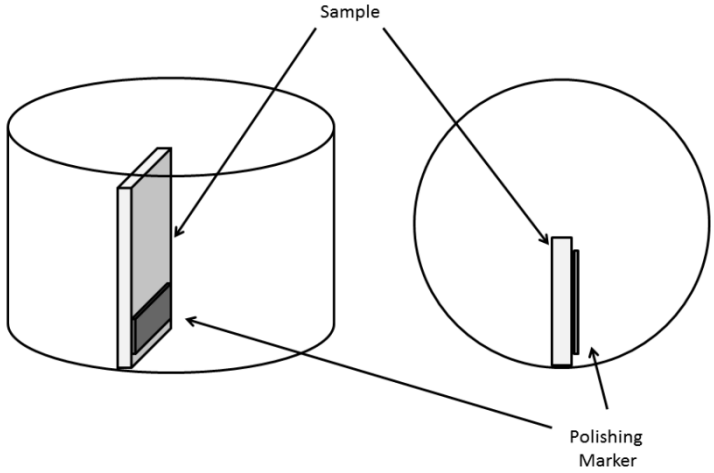

Figure 2.2: Three-dimensional visualization of the puck including the polishing marker looking from the side (left) and bottom (right).

2.2.1.4 Sample Grinding

After the sample has been mounted the first of a two-step procedure for obtaining a smooth surface of which to image can be carried out. Sample grinding is the use of sand paper to gradually wear down the bottom of the puck until the marker on the sample is reached. During the grinding procedure it is important to keep the surface of the sample wet to avoid fiber breakage. Rotation of the sample is also recommended to evenly wear down the sample instead of causing local height fluctuations. A summary of the general procedure for grinding the resins containing the samples of fiber filled thermoplastic is given in Table 2.1. The first "stage" uses a welt-belt grinder (BG-32 dual belt grinder; LECO) and removes material quickly but must be done with care so as to not damage the sample in question. The wet-belt grinder is used twice with both 120-grit and 180-grit, done in succession, to remove material the desired amount of material. The second stage of grinding is done by hand and uses a wet hand grinder (Handimet® 2 Roll Grinder; Buehler Co.) and uses the times and conditions outlined in Table 2.2. The use of the two stages outlined in this section should be enough to display the plane of interest of the sample in the resin puck.

Stage	CAMI Grit Designation (Average sizes of the abrasive material in μm)	Time (min)	Interval for Rotation (s)	Objective
I (Wet Belt Grinder)	120 (115)	1-2 (Until Marker is seen, blurry under surface)	10	Removal of material until plane of interest is reached
	180 (82)	1-2 (Until Marker is seen, clear surface)	10	
II (Wet Hand Grinder)	240 (53)	2	30	Removal of material to eliminate surface grooves
	320 (36)	2	30	
	400 (23)	2	30	
	600 (16)	2	30	

Table 2.1: Detailed procedure for the grinding portion of the experimental sample preparation adapted from [21].

The method of ellipses (discussed in § 2.2.3) is highly dependent on the sampling surface being completely flat. To quantify the surface height of the sample, and to make sure that the plane of interest is flat, measurements of the surface height are taken throughout the grinding process at four 90 degree increments around the puck (0°, 90°, 180°, 270°). These heights are monitored during the entire process of grinding and polishing. Careful rotation and height monitoring yield surface height fluctuations of less than 0.010 mm [21].

2.2.1.5 Sample Polishing

The polishing of samples is the other of the important two step processes for obtaining smooth sample surfaces to image. An automated grinder (MiniMet® 1000 grinder/polisher; Buehler Co.) is used to carry out the operation and is crucial for providing the necessary control needed for obtaining polished sample surfaces of the utmost quality. Holes are drilled in the top center of the puck so that they can be mounted on the apparatus. Synthetic nap cloth (Final finish cloth; Precision Surfaces International, Inc.) are used to hold alumina particles of various sizes (Precision Surfaces International, Inc.) in an aqueous suspension. The details of the polishing procedure are summarized in Table 2.2.

Alumina Particle Size (μm)	Speed (RPM)	Vertical Force (N)	Time (min)
5	40-50	22.2	40
1	40-50	22.2	40
0.3	40-50	22.2	30
0.05	40-50	22.2	30

Table 2.2: Detailed procedure for the polishing portion of the experimental sample preparation adapted from [21].

The quality of the sample surface is related to not only the height fluctuations of the sample but also the quality of ellipses seen on the sample surface. The method of ellipses (discussed in §2.2.3) needs elliptical footprints of fiber cross-sections in order to make quantitative arguments on the orientation of fibers within a sample.

2.2.1.6 Plasma Etching

In image processing, and especially in automated image recognition, the color contrast between the object of interest and the background is extremely important [22]. For some polymer matrices containing glass fibers, polishing alone does not provide a large enough contrast. Plasma etching is one of the few experimental techniques that can be used to increase the contrast between object and background. Oxygen plasma is generated under vacuum to roughen the polished surface of the sample, burning away the matrix and leaving the fibers.

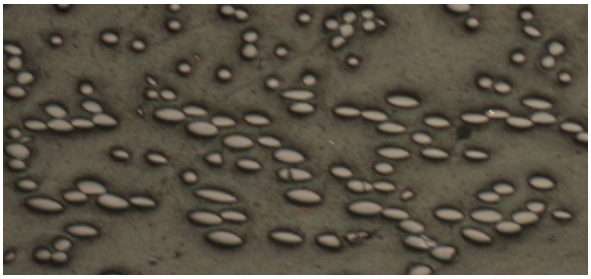

Figure 2.3: Image showing final stage of sample preparation once sample has been etched using oxygen plasma under 10 x magnification.

The polished surface becomes darker during this processes leading to a larger color gradient between matrix and fiber. Junkar et al. [23] have shown that etching times vary according to matrix and is related to the crystallinity of the particular polymer.

2.2.2 Image Acquisition

With a large color gradient between the fibers and surrounding matrix, it is now possible to acquire images of samples and generate experimental orientation data. Clarke and Eberhardt [24] argue that 20x magnification of the sample provides adequate resolution. An optical – reflection microscope (Axiovert 40 MAT; Carl Zeiss LLC) is used to acquire images. The use of a motorized stage is also employed so that multiple images can be acquired and "stitched" together to form a single large image.

2.2.3 Image Processing and Experimental Fiber Orientation Calculation

Experimental fiber orientation is determined using the "Method of Ellipses" which is a quasi-three-dimensional method for calculating the orientation of a fiber within a sample. The fiber orientation can be calculated with the knowledge of two angles, θ and ϕ, shown in Figure 2.4.

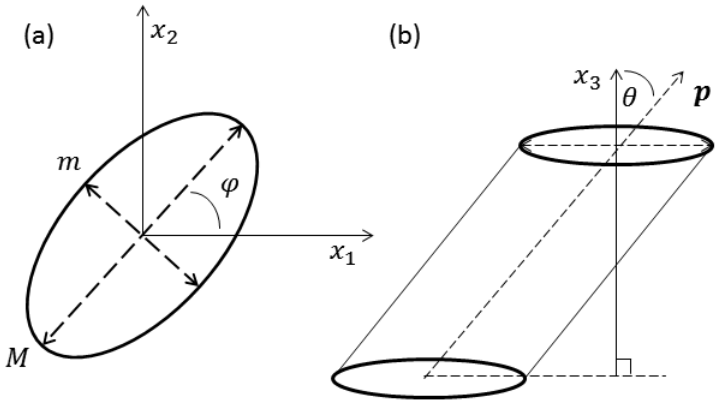

Figure 2.4: Determination of experimental fiber orientation using: (a) in-plane view where M is the length of the major axis, m is the length of the minor axis, and j is the in-plane angle, (b) out-of-plane view where $\mathbf{p}$ is the vector projected down the major axis of the fiber and q is the out-of-plane angle.

Error! Reference source not found. (a) shows the in-plane ellipse that is commonly encountered in the experimentally obtained images, similar to that shown in Error! Reference source not found.. The in-plane angle is calculated using simple trigonometry. Error! Reference source not found. (b) shows the out of plane angle, or the "projection" of the fiber into and out of the plane. The out-of-plane angle is then calculated with Eq. (2.7)**:**

$$\theta = \cos^{-1}\left(b/a\right) \tag{2.7}$$

2.2.4 Summary of Experimental Fiber Orientation

The use of the experimental technique above makes assumptions about both the length and flexibility of the glass fibers in the system under study. The method of ellipses was developed for rigid SGF systems so that the original choice of a 0.7 mm bin-width would be appropriate for a system whose average fiber length is around 0.3 mm (a bin width roughly twice the size of the fiber being investigated). In a LGF system, it is possible to have average fiber

lengths around 3.5 mm or greater and single fibers reaching lengths of 10 mm. The bending of these longer fibers is also commonly seen during image processing. With longer fibers it is easy to see that the traditional bin width of 0.7 mm and the assumption that they are rigid is inappropriate.

At the time of this work two bin widths of 0.7 mm and 5.5 mm were adopted to account for the longer fibers in the system. The two bind widths are being studied to determine the effect that each have of experimental fiber orientation. At the time of this work, no work has been performed to quantify the orientation of a fiber which is bent.

2.3 Modeling Equations for Stress Tensors in Fiber Suspensions

In the prediction of glass fiber orientation, parameters arise which are fitted using equations of stress for fiber suspensions. In the following section, the development of stress tensor for rigid and semi-flexible fiber systems is presented. First, the development of a stress tensor for rigid fiber filled systems is discussed in § 2.3.1. Then, the development of a stress tensor for flexible fiber systems is covered in § 2.3.2.

2.3.1 Stress Tensor Modeling in Rigid Fiber Suspensions

As described in § 2.1.1, fiber suspensions can be classified by concentration. For the development of a stress tensor two distinctions are made using the definitions from § 2.1.1: *dilute/semi-concentrated* and *concentrated*. The stresses inherent to a glass fiber composite differ greatly based on a number of factors including fiber volume fraction, fiber length and fiber orientation to name a few.

2.3.1.1 Model for the Stress Tensor in Rigid Dilute Suspensions

Modeling the stress tensor and viscosity changes that occur with additives to solutions have been studied since the work by Einstein [25] where the relative viscosity, η_r, of a dilute dispersion of small spheres was modeled as Eq. (2.8) and based on the volume fraction of the particles, ϕ:

$$\eta_r = 1 + 2.5\phi \tag{2.8}$$

Further analysis by Mooney [26] was performed where Eq. (2.9) was developed for more concentrated systems where particle "crowding" was taken into account (represented by k) and is the first known relation where interactions are included with the viscosity relation.

$$\eta_r = \exp\left\{\frac{2.5\phi}{1-k\phi}\right\} \tag{2.9}$$

Hand [27], Giesekus [28, 29] and Batchelor [30] presented a method using averaged values over a collection of suspended particles using orientation tensors. The most general form of the stress equation is given by Eq. (2.10) which gives a full account of interparticle interaction where a, b, c, and f are geometric shape factors based on the specific system in question [31]:

$$\bar{\bar{\sigma}} = -P\bar{\bar{I}} + 2\eta_m\bar{\bar{D}} + 2\eta_m\phi\left[a\bar{\bar{\mathbb{A}}}_4 : \bar{\bar{D}} + b\left(\bar{\bar{D}}\cdot\bar{\bar{A}} + \bar{\bar{A}}\cdot\bar{\bar{D}}\right) + c\bar{\bar{D}} + fd_R\bar{\bar{A}}\right] \tag{2.10}$$

In Eq. (2.10) $\bar{\bar{\sigma}}$ is the total stress tensor, P is the isotropic stress, $\bar{\bar{I}}$ is the unit tensor, η_m is the matrix viscosity, $\bar{\bar{D}}$ is the rate of deformation tensor, ϕ is the volume fraction of fibers in the system, $\mathbb{A}_4$ is the 4th moment of the orientation distribution function, $\bar{\bar{A}}$ is the 2nd moment of orientation distribution function. d_r is a term which tries to account for Brownian motion within the system, however, for LGF systems the fibers are considered to be non-Brownian (see § **B**.1) so this term may be discarded. Similarly, for long aspect ratio fibers, b and c are small in comparison to other terms so these are also neglected. The final equation is thus reduced to a form for fiber suspensions shown in Eq. (2.11) where N attempts to represent the associated anisotropic stresses associated with the system [31]:

$$\bar{\bar{\sigma}} = -P\bar{\bar{I}} + 2\eta_m\bar{\bar{D}} + 2\eta_m\phi\left(N\bar{\bar{\mathbb{A}}}_4 : \bar{\bar{D}} + \bar{\bar{D}}\right) \tag{2.11}$$

Batchelor [30] proposed a similar equation to account for particle-fluid and particle-particle interactions, shown in Eq. (2.12):

$$\bar{\bar{\sigma}} = -P\bar{\bar{I}} + 2\eta_m\left(\bar{\bar{D}} + N\bar{\bar{\mathbb{A}}}_4 : \bar{\bar{D}} - \tfrac{1}{3}N\overline{\overline{AI}} : \bar{\bar{D}}\right) \tag{2.12}$$

Estimations have been proposed for the value of N for a dilute suspension of particles, most notably by Batchelor [32] where n is the number of fibers per unit volume, L is the length of the fiber, d is the fiber diameter, and $\varepsilon = 1/\ln(2L/d)$.

$$N = \frac{\pi n L^3}{6\ln\left(2L/d\right)}\left(\frac{1+0.64\varepsilon}{1-1.5\varepsilon} + 1.659\varepsilon^2\right) \tag{2.13}$$

Corrections to dilute theory for two body interactions were proposed by Shaqfeh and Fredrickson [33] and are shown for aligned suspensions and isotropic suspensions in Eq. (2.14) and Eq. (2.15), respectfully:

$$N = N^{dilute} + 8.583\times10^{-3}\frac{\pi\left(nL^3\right)^2}{\ln\left(2L/d\right)^3} \tag{2.14}$$

$$N = N^{dilute} + 9.250\times10^{-3}\frac{\pi\left(nL^3\right)^2}{\ln\left(2L/d\right)^3} \tag{2.15}$$

For semi-dilute systems (see § 2.1.1 (b)), Dinh and Armstrong [5] provided a different relation for N seen in Eq. (2.16):

$$N = \frac{\pi n L^3}{6\ln\left(2h/d\right)} \tag{2.16}$$

Where h is defined for specific systems as either random or aligned in Eq. (2.17) and Eq. (2.18), respectively:

$$h = \frac{1}{nL^2} \tag{2.17}$$

$$h = \frac{1}{(nL)^{1/2}} \tag{2.18}$$

An expression for h has also be presented for use in systems that are a combination of the two above [31]. While many of these relations have seen some success in modeling dilute and semi-dilute systems, all break down when attempting to predict concentrated fiber suspensions.

2.3.1.2 Model for the Stress Tensor in Rigid Non-Dilute Suspensions

Stresses in non-dilute suspensions are generally treated as linear combinations of the matrix and particle contributions. Early work was performed by Ericksen [34], Batchelor [32], Goddard [35] and Lipscomb [36] in modeling the stress tensor in a concentrated suspension. Lipscomb [36] presented a theory which included the fiber volume fraction, ϕ, in the stress equation, seen in Eq. (2.19):

$$\bar{\bar{\sigma}} = -P\bar{\bar{I}} + 2\eta_m\left(\bar{\bar{D}} + c_1\phi\bar{\bar{D}} + N\bar{\bar{\mathbb{A}}}_4 : \bar{\bar{D}}\right) \tag{2.19}$$

The value of N is present to account for the effect fiber orientation has on the stress enhancement of the system while the value of c_1 attempts to quantify the stress enhancement of the amount of fiber in the system. Eberle et al. [37] have chosen to use N as a fitting parameter and the value of c_1 is often fit as well [10]. Varying degrees of success have been reported by Ortman et al. [38] using this technique depending on the initial fiber orientation of the system.

Ait-kadi and Grmela [39] and Azaiez [40] developed viscoelastic models to predict stresses in fiber suspensions. The composite stress is given in Eq. (2.20) with the Giesekus model for the polymer behavior following in Eq. (2.21):

$$\bar{\bar{\tau}}_c = \eta_m\kappa\bar{\bar{\dot{\gamma}}} + \eta_m\left[\kappa + \frac{\eta(\dot{\gamma})}{\eta_m}\right]\phi\left[c_1\bar{\bar{\dot{\gamma}}} + c_2\bar{\bar{\dot{\gamma}}} : \bar{\bar{\mathbb{A}}}_4\right] + \bar{\bar{\tau}}_p \tag{2.20}$$

$$\lambda\bar{\bar{\tau}}_{p(1)} + \sigma\bar{\bar{\tau}}_p - \frac{\alpha\lambda}{\eta_m}\left(\bar{\bar{\tau}}_p\cdot\bar{\bar{\tau}}_p\right) + \frac{m(1-\sigma)}{2}\left(\bar{\bar{A}}\cdot\bar{\bar{\tau}}_p + \bar{\bar{\tau}}_p\cdot\bar{\bar{A}}\right) = -\eta_m\bar{\bar{\dot{\gamma}}} \tag{2.21}$$

Other modeling techniques have been suggested for predicting stresses in fiber suspensions [39]. The use of viscoelastic or other complicated models to calculate a composite stress tensor may increase accuracy but also complicate the solution of a problem by presenting yet another set of coupled differential equations to solve decreasing the stability of the solution.

2.3.2 Stress Tensor Modeling in Flexible Fiber Suspensions

The flexibility of a fiber was introduced in § 2.1.2 as a way of quantifying fiber bending in a flow field. Forgacs and Mason [41] investigated the role of flexibility in fiber rotations and also found a "critical bending stress" for a single fiber in dilute solution. This relation is given in Eq. (2.22) where E_b is the bending modulus $\left(E_b = 2E_Y\right)$ and a_r is the aspect ratio.

$$\left(\dot{\gamma}\eta_m\right)_{\text{crit}} \cong \frac{E_b\left(\ln 2a_r - 1.75\right)}{2a_r^4} \tag{2.22}$$

This relation shows that axial forces caused by shear flow can bend fibers of high aspect ratio or low modulus. Goto et al. [42] carried out a number of studies using glass fibers in both a Newtonian and non-Newtonian medium. It was shown that in both cases, fiber flexibility greatly increased both the viscous and elastic properties of the fluid.

2.3.2.1 Model for the Stress Tensor in Dilute Flexible Fiber Suspensions

The calculation of a stress tensor for a dilute suspension containing flexible or semi-flexible fibers has been studied mainly by direct simulation [12, 43-48]. These simulations require detailed information about the physical interactions on a *micro* or *meso* scale in the system and are often very computationally intensive. Because of the laborious computational technique, the number of fibers must often be limited, thus reducing accuracy for a large system of fibers and yielding different theories about the effect that fiber flexibility has on a suspension. Joung et al. [38] directly simulated the physics of fibers interacting in a Newtonian fluid, using a method where the flexible fibers were modeled as chains of beads joined by connectors, and showed that the suspension viscosity increased with increased fiber flexibility. Switzer and Klingenberg [42] carried out simulations modeling the fibers as linked cylinders with ball-and-socket joints and showed that suspension viscosity decreased with increasing fiber flexibility. Rajabian et al. [49] instead used an entropy based approach to predict the stress tensor but no experimental data has been published corroborating the results.

The contradiction of results led to a thorough investigation by Keshtar et al. [3] where a number of fibers of different flexibilities were investigated in steady and transient shear flows in both dilute and semi-dilute concentration. Increasing the flexibility of the fiber enhanced the shear-thinning behavior of the suspension. In start-up flow, large transient over shoots were reported with the magnitude of overshoot increasing with flexibility, which is also consistent with the results reported for rigid fiber suspensions [50]. The prediction of the stress tensor in the fiber suspensions investigated by Keshtar et al. [3, 50] was performed using the general equation for non-equilibrium reversible-irreversible coupling (GENERIC) which was first proposed by Grmela and Öttinger [51] and Öttinger and Grmela [52]. Here, the modeling utilizes the orientation tensors for fiber orientation (see § 2.4.1) along with a normalized end-to-end vector of the fiber which can be seen in Figure 2.7. The GENERIC model is based on the entropy within a system. The GENERIC model is advantageous only in that it is always consistent with the hydrodynamic and thermodynamic effects. The major disadvantage is the lack of connection between the fiber dynamics and the empirical parameters in to calculate them. Specifically, free energy and dissipation parameters directly determine the fiber dynamics of the system and are

not measurable functions of the system in question. These inconsistencies lead to consistent overshoots in the prediction of the transient stresses [53].

2.3.2.2 Model for the Stress Tensor in Concentrated Flexible Fiber Suspensions

Very little work has been reported in predicting the stress tensor in concentrated flexible fiber suspensions. Ortman et al. [54] has reported some success modeling concentrated semi-fiber suspensions using a combination of the theory presented by Strautins and Latz [55] for flexible dilute systems and the theory presented by Dinh and Armstrong [5] for semi-dilute fiber suspensions. The stress tensor is presented in Eq. (2.23)where a 3rd term has been added to Eq. (2.19) to account for fiber flexibility. In Eq. (2.23), c_3 is an empirical fitting parameter, k a flexibility parameter for the fiber material, $\bar{r}$ is the end-to-end vector (see §), l_b is half of the average fiber length, and $\bar{\bar{R}}$ is the normalized 2nd moment of the end-to-end tensor or $\bar{\bar{R}} = \left(\bar{\bar{A}} - \bar{\bar{B}}\right) \Big/ \left(1\text{-tr}\left(\bar{\bar{B}}\right)\right)$:

$$\bar{\bar{\sigma}}_{flex} = -P\bar{\bar{I}} + 2\eta_m\left(\bar{\bar{D}} + f_1\phi\bar{\bar{D}} + f_2\bar{\bar{\mathbb{A}}}_4 : \bar{\bar{D}}\right) + c_3\eta_m k\frac{3\phi a_r}{2}\frac{\text{tr}\left(\bar{r}\right)}{2l_b^2}\left(\bar{\bar{A}} - \bar{\bar{R}}\right) \tag{2.23}$$

Ortman et al. [54] provided a quantitative argument for the parameters given in Eq. (2.24) and Eq. (2.25)for f_1 and f_2, respectfully:

$$f_1 = \begin{cases} \dfrac{c}{\dot{\gamma}^{\min}} & \dot{\gamma} \le \dot{\gamma}^{\min} \\ \dfrac{c}{\dot{\gamma}} & \dot{\gamma} > \dot{\gamma}^{\min} \end{cases} \tag{2.24}$$

$$f_2 = c_2 I_A II_A III_A \tag{2.25}$$

where invariants of the orientation tensor, $\bar{\bar{A}}$, are given in Eqs. (2.26) to (2.28):

$$I_{\mathbf{A}} = \text{tr}\left(\bar{\bar{A}}\right) \tag{2.26}$$

$$II_{\mathbf{A}} = \frac{1}{2}\left[\text{tr}\left(\bar{\bar{A}}\right)^2 - \text{tr}\left(\bar{\bar{A}}\bar{\bar{A}}\right)\right] \tag{2.27}$$

$$III_{\mathbf{A}} = \det\left(\bar{\bar{A}}\right) \tag{2.28}$$

Using the stress model presented in Eq. (2.23) and choosing to make the parameters f_1 and f_2 a function of shear rate and fiber orientation has been shown to be quantitatively accurate [56]. To date the method presented by Ortman et al. [57]provides the highest amount of accuracy with relatively little computation time.

2.3.3 Summary of Stress Tensor Modeling

The methods presented in the preceding section outline techniques for modeling the stress tensor of a suspension. It can be seen from the presented literature that the modeling of dilute solutions is difficult and relatively little is understood from the perspective of first principles. This misunderstanding is magnified when trying to describe increased fiber loading and fiber-fiber collision because even the most advanced modeling efforts still have empirical parameters that must be fit.

Further complicating the matter, long glass fibers have been shown to exhibit flexing in simple shear flow experiments. Ortman et al. [38] showed that modeling the stress tensor of a long glass fiber suspension was possible through the fitting of the concentrated rigid stress model, Eq. (2.19)but still relied on empirical fitting transient stress curves. At the time of this review, no other theory for predicting the stress tensor of long glass fiber systems has been presented which accounts for the additional bending that may occur.

2.4 Modeling Equations for Fiber Orientation

The probability that a fiber will be oriented between the angles (θ_1,ϕ_1) and $(\theta_1+d\theta,\phi_1+d\phi)$, is given by Eq. (2.29):

$$P(\theta_1 \le \theta \le \theta_1+d\theta,\phi_1 \le \phi \le \phi_1+d\phi) = \psi(\theta_1,\phi_1)\sin\theta_1 \; d\theta \; d\phi \tag{2.29}$$

Some authors have used the probability distribution function to describe the orientation in fiber systems [58], but it has been observed that the addition of too many fibers to the system of interest can make the calculations cumbersome [59]. The use of orientation tensors, given by Advani and Tucker [1], will instead be used and is widely accepted to be an accurate representation of fiber orientation within a sample.

The following section is divided up into two main sections for examining rigid fiber orientation models and semi-flexible fiber orientation models. The rigid fiber orientation models are examined in § 2.4.1 for both the dilute and concentrated case. The semi-flexible fiber models for orientation are reviewed in § 2.4.2.

2.4.1 Rigid Fiber Orientation

Defining a rigid fiber's particular orientation is done by representing that fiber as a vector running through the longitudinal axis seen in Figure 2.5.

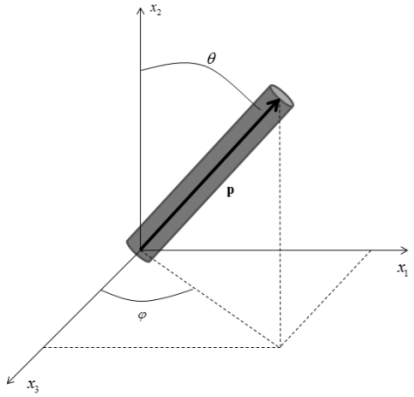

Figure 2.5: Rigid-rod representation of fiber. The fiber is represented by the vector $\overline{p}$, and is characterized by the azimuthal and zenith angles j and q, respectfully.

A compact method for quantifying fiber orientation is through the use of orientation tensors. This method was explored by Erickson [34, 60, 61] for liquid crystalline polymers and later by Advani and Tucker [62] for representing rigid fiber orientation. Advani and Tucker [62] described the second and fourth moments of the orientation distribution function by Eq. (2.30) and Eq. (2.31), respectively:

$$\overline{\overline{A}} = \iint \overline{pp}\psi(\overline{p},t)d\overline{p} \tag{2.30}$$

$$\overline{\overline{\mathbb{A}}}_4 = \iint \overline{pppp}\psi(\overline{p},t)d\overline{p} \tag{2.31}$$

An issue arises with the calculation of Eq. (2.31) as there is no exact way to calculate the fourth moment of the orientation distribution function. Instead the fourth moment must be represented as a combination of second moments of the orientation distribution function. The estimations are generally referred to as closure approximations and are the subject of a number of researcher's efforts [31]. The choice of closure models is a significant part of the numerical simulation scheme and will be the subject of discussion later (§ 2.4.4).

2.4.1.1 Jeffery's Equation (Dilute Rigid Fiber Suspensions)

The motion of a single ellipsoidal particle, proposed by Jeffery [63], can be represented by Eq. (2.32) where $\overline{p}$ is the directional vector of a single rigid ellipsoid, $\overline{v}$ is the velocity vector, $\overline{\overline{W}}$ is the vorticity $\left(\overline{\overline{W}} = \frac{1}{2}\left[\nabla\overline{v} - (\nabla\overline{v})^T\right]\right)$, $\overline{\overline{D}}$ is the rate of deformation $\left(\overline{\overline{D}} = \frac{1}{2}\left[\nabla\overline{v} + (\nabla\overline{v})^T\right]\right)$, and ξ is the shape factor $\xi = a_r^2 - 1/a_r^2 + 1$.

$$\frac{\partial\overline{p}}{\partial t} + \mathbf{v}\cdot\nabla\overline{p} = \overline{\overline{W}}\cdot\overline{p} + \xi\left[\overline{\overline{D}}\cdot\overline{p} - \overline{p}\left(\overline{p}\cdot\overline{\overline{D}}\cdot\overline{p}\right)\right] \tag{2.32}$$

Jeffery predicted that the particles, in laminar flow, would undergo periodic rotations in a spherical elliptical orbit experimentally confirmed by Trevelyan and Mason [64]. In simple shear

flow $\left(v_1 = \dot{\gamma} y,\ v_2 = v_3 = 0\right)$, the differential equations are exact and are written in Eq. (2.33) and Eq. (2.34) .

$$\frac{\partial \phi}{\partial t} = \frac{\dot{\gamma}}{a_r^2 + 1}\left(a_r^2 \cos^2 \phi + \sin^2 \phi\right) \tag{2.33}$$

$$\frac{\partial \theta}{\partial t} = \xi \sin 2\theta \sin 2\phi \tag{2.34}$$

The solution is easily solved for and is shown in Eq. (2.35) through Eq.(2.38).

$$\tan \phi = a_r \left(\frac{\dot{\gamma} t}{a_r + a_r^{-1}} + k \right) \tag{2.35}$$

$$\tan k = \left(\frac{1}{a_r \tan \phi_0} \right) \tag{2.36}$$

$$\tan \theta = \frac{C a_r}{\sqrt{a_r^2 \sin^2 \phi + \cos^2 \phi}} \tag{2.37}$$

$$C = \frac{\tan \theta_0}{\sqrt{a_r^2 \sin^2 k + \cos^2 k}} \tag{2.38}$$

Where k and C are obtained from the initial conditions of the spheroid, ϕ_0 and θ_0. The solution to the above equations is periodic and is described by Eq. (2.39) with Figure 2.6 showing the periodic rotations for a specific case of $a_r = 5$ and initial conditions $\phi_0 = 175°, \theta_0 = 45°$. In Eq. (2.39) T is the period of rotation, $\dot{\gamma}$ is the magnitude of the shear rate and a_r is the aspect ratio of the particle.

$$T = \frac{2\pi}{\dot{\gamma}} \left(a_r + \frac{1}{a_r} \right) \tag{2.39}$$

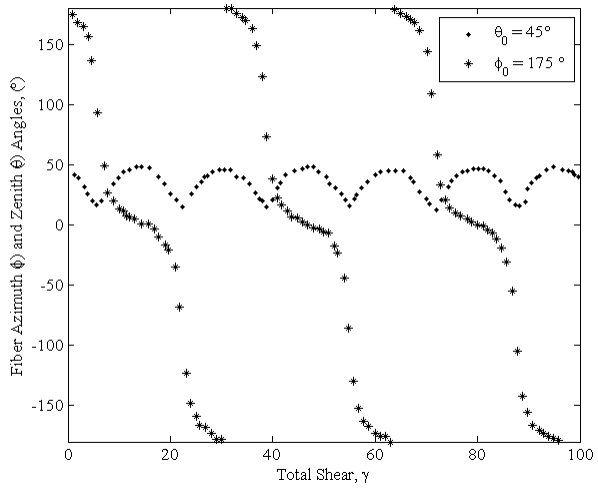

Figure 2.6: Polar and azimuthal angles describing orientation kinetics of an ellipsoidal particle in simple shear flow given by the analytical solution to Jeffery's equation, Eq. (2.35) to Eq. (2.38). Solution is given for a particle with $a_r = 5$ and initial conditions $\phi_0 = 175°, \theta_0 = 45°$.

Further analyses of Jeffery's orbits for different cases have been performed by Barbosa and Bibbo [65]. Mason and Manley [66] addressed the quantitative periodic rotation of rigid rods with distribution functions to measure orientation of rods within a desired coordinate plane. Furthermore, Mason and Manley documented drift from one Jeffery's orbit to another showing that orientation kinetics are significantly more complicated than Jeffery first estimated. Collisions between two particles were further studied by Manley and Mason [67]. Ferguson [68] addressed the issue of rotating rigid particles in a second-order Rivlen-Erickson fluid and confirmed that the particles slowly drift through a number of different orbits over time.

A continuum form of Jeffery's equation for a single spheroid can be obtained by applying Eq. (2.30) and Eq. (2.31) to the model in Eq. (2.32). The left-hand side of Eq. (2.40) is the material derivative and will be written as the more compacted form, $\frac{D\overline{\overline{A}}}{Dt} = \frac{\partial \overline{\overline{A}}}{\partial t} + \overline{v} \cdot \nabla \overline{\overline{A}}$, throughout this chapter.

$$\frac{D\overline{\overline{A}}}{Dt} = \overline{\overline{W}} \cdot \overline{\overline{A}} - \overline{\overline{A}} \cdot \overline{\overline{W}} + \xi\left(\overline{\overline{D}} \cdot \overline{\overline{A}} + \overline{\overline{A}} \cdot \overline{\overline{D}} - 2\overline{\overline{D}} : \overline{\overline{\mathbb{A}}}_4\right) \tag{2.40}$$

In the equation above, $\overline{\overline{A}}$ is the second moment of the orientation tensor, $\mathbb{A}_4$ is the fourth moment of the orientation tensor, $\overline{\overline{W}}$ is the vorticity tensor $\left(\overline{\overline{W}} = \frac{1}{2}\left[\nabla\overline{v} - (\nabla\overline{v})^T\right]\right)$, $\overline{\overline{D}}$ is the rate or deformation tensor $\left(\overline{\overline{D}} = \frac{1}{2}\left[\nabla\overline{v} + (\nabla\overline{v})^T\right]\right)$ and ξ is a shape factor. For glass fiber systems the

shape factor is usually given a value of unity, $\xi \to 1$. Objectivity has been attained thus far for Eq. (2.40) as the equation is in a form containing the co-deformational derivative of $\overline{\overline{A}}$.

2.4.1.2 Concentrated Rigid Fiber Suspensions

The addition of an isotropic rotary diffusion term to Eq. (2.40) was suggested by Folgar and Tucker [69] to more accurately describe the orientation kinetics exhibited by concentrated fiber systems and is shown in Eq. (2.41) where $\dot{\gamma}$ is the magnitude of the shear rate, C_I is a phenomenological constant and $\overline{\overline{I}}$ is the identity tensor:

$$\frac{D\overline{\overline{A}}}{Dt} = \overline{\overline{W}} \cdot \overline{\overline{A}} - \overline{\overline{A}} \cdot \overline{\overline{W}} + \xi\left(\overline{\overline{D}} \cdot \overline{\overline{A}} + \overline{\overline{A}} \cdot \overline{\overline{D}} - 2\overline{\overline{D}} : \overline{\overline{\mathbb{A}}}_4\right) + 2\dot{\gamma}C_I\left(\overline{\overline{I}} - 3\overline{\overline{A}}\right) \tag{2.41}$$

Folgar and Tucker [69] theorized that in concentrated suspensions fiber-fiber interactions could be captured by the additional term. The model was shown to agree well with experimental data from a center-gated disk for short glass fiber systems by Bay and Tucker [70] at 10% of the cavity fill but over-predicted wall orientation at 40% at 90% of the mold fill.

The fiber interaction coefficient, C_I, has been continuously studied to improve fiber orientation predictions. Based on a number of experiments Bay [71] suggested Eq. (2.42) to empirically model the fiber interaction where ϕ is the fiber volume fraction and a_r is the aspect ratio:

$$C_I = 0.0184 \exp\left\{-0.7148\phi a_r\right\} \tag{2.42}$$

Ranganathan and Advani [72] proposed that the fiber interaction coefficient inversely proportional to the average distance between fibers. Phan-Thien et al. [73] proposed a fiber interaction coefficient in the form of Eq. (2.43) where M and N were obtained empirically through simple shear experiments as 0.03 and 0.224, respectively:

$$C_I = M\left[1.0 - \exp\left\{-N\phi a_r\right\}\right] \tag{2.43}$$

Stress growth experiments in the startup of simple shear suggested that the orientation kinetics in concentrated fiber systems evolved slower than Eq. (2.41) predicted. A slip parameter was suggested to delay the predicted fiber orientation to better agree with experimental orientation data [7, 74], seen in Eq. (2.44):

$$\frac{D\overline{\overline{A}}}{Dt} = \alpha\left[\overline{\overline{W}} \cdot \overline{\overline{A}} - \overline{\overline{A}} \cdot \overline{\overline{W}} + \xi\left(\overline{\overline{D}} \cdot \overline{\overline{A}} + \overline{\overline{A}} \cdot \overline{\overline{D}} - 2\overline{\overline{D}} : \overline{\overline{\mathbb{A}}}_4\right) + 2\dot{\gamma}C_I\left(\overline{\overline{I}} - 3\overline{\overline{A}}\right)\right] \tag{2.44}$$

The "slip" parameter, α, has a value between 0 and 1 and slows the fiber orientation predictions to better match experimental conditions. The addition of this coefficient does result in a loss of material objectivity [75] but is still useful in predicting experiment fiber orientation.

To model the slow orientation kinetics of glass fiber filled systems and still maintain objectivity Wang et al. [76] proposed the "Reduced Strain Closure" (RSC) whereby the slip parameter was moved into the double dot product of Eq. (2.44) producing Eq. (2.45). It is important to note that a closure approximation is still required for the fourth-order moment orientation tensor, $\mathbb{A}_4$, in Eq. (2.45):

$$\frac{D\overline{\overline{A}}}{Dt} = \overline{\overline{W}}\cdot\overline{\overline{A}} - \overline{\overline{A}}\cdot\overline{\overline{W}} + \xi\left\{\overline{\overline{D}}\cdot\overline{\overline{A}} + \overline{\overline{A}}\cdot\overline{\overline{D}} - 2\left[\overline{\overline{\mathbb{A}}}_4 + (1-\kappa)\left(\overline{\overline{\mathbb{L}}}_4 - \overline{\overline{\mathbb{M}}}_4 : \overline{\overline{\mathbb{A}}}_4\right) : \overline{\overline{D}}\right]\right\} + 2\kappa\dot{\gamma}C_I\left(\overline{\overline{I}} - 3\overline{\overline{A}}\right) \tag{2.45}$$

In Eq. (2.45) κ is a scaling factor between 0 and 1, $\overline{\overline{\mathbb{A}}}_4$ is the fourth-order moment orientation tensor, and $\overline{\overline{\mathbb{L}}}_4$ and $\overline{\overline{\mathbb{M}}}_4$ are the corresponding fourth-order moment eigenvalue and eigenvector tensors defined in Eq. (2.46) and Eq. (2.47), respectfully:

$$\overline{\overline{\mathbb{L}}}_4 = \sum_{i=1}^{3} \lambda_i \overline{e}_i \overline{e}_i \overline{e}_i \overline{e}_i \tag{2.46}$$

$$\overline{\overline{\mathbb{M}}}_4 = \sum_{i=1}^{3} \overline{e}_i \overline{e}_i \overline{e}_i \overline{e}_i \tag{2.47}$$

The RSC model has not been tested in systems containing long glass fibers so the accuracy of such a model cannot be made. The reference frame for predicting fiber orientation during injection molding does not change (there is no rotation or translation of the coordinate frame) so while Eq. (2.45) does maintain objectivity it is ultimately unnecessary from an industrial modeling standpoint.

The isotropic rotary diffusion term, first introduced in Eq. (2.41), is a simplification for materials that exhibit isotropy, or uniformity in all directions. In commercial application of glass fibers, systems are rarely isotropic. The anisotropic nature of these materials was first accounted for by Koch [77] where the diffusion constant, C_I, was instead represented by a tensor quantity, $\mathbf{C}$. With the addition of this tensor, the fiber orientation equations could be defined as a combination of hydrodynamic and diffusive contributions, shown in Eq. (2.48). The hydrodynamic contribution is unchanged from the original derivation from Jeffery's equation and takes the form of Eq. (2.40) .

$$\frac{D\overline{\overline{A}}}{Dt} = \overline{\overline{A}}^h + \overline{\overline{A}}^d \tag{2.48}$$

Koch [77] defined a fiber mobility tensor as Eq. (2.49) where n is the number of fibers per unit volume and $\mathfrak{A}$ is the sixth moment orientation tensor shown in Eq. (2.50).

$$\overline{\overline{C}} = \frac{nL^3}{\dot{\gamma}^2 \ln\left(a_r\right)^2}\left[\beta_1\left(\overline{\overline{D}} : \overline{\overline{\mathbb{A}}}_4 : \overline{\overline{D}}\right)\overline{\overline{I}} + \beta_2 \overline{\overline{D}} : \overline{\overline{\mathfrak{A}}}_6 : \overline{\overline{D}}\right] \tag{2.49}$$

$$\overline{\overline{\mathfrak{A}}}_6 = \iint \overline{pppppp}\psi(\overline{p},t)d\overline{p} \tag{2.50}$$

The sixth moment tensor introduces the need for a different closure approximation in terms of the second moment tensor, however, a closure a simple as Eq. (2.74) is not viable. Sixth-order closures are briefly discussed in § 2.4.4.

Koch's development of the fiber mobility tensor gave a diffusion contribution for Eq. (2.48) as Eq. (2.51), but the theory was based on a semi-dilute suspension and gives values above and below unity when predicting fiber orientation in concentrated suspensions as well as predicting non-symmetric second moment orientation tensors:

$$\overline{\overline{A}}^d = \dot{\gamma}\left[2\overline{\overline{C}} - 2\ \mathrm{tr}\left(\overline{\overline{C}}\right)\overline{\overline{A}} - 5\left(\overline{\overline{C}}\cdot\overline{\overline{A}} + \overline{\overline{A}}\cdot\overline{\overline{C}}\right) + 10\overline{\overline{\mathbb{A}}}_4 : \overline{\overline{C}}\right] \tag{2.51}$$

Fan et al. [78] and Phan-Thien et al. [73] corrected the problems stated above by proposing Eq. (2.52) but using their model yields $\overline{\overline{A}}^d \neq 0$ for an isotropic system which violates the conservation of diffusive flux:

$$\overline{\overline{A}}^d = \dot{\gamma}\left[2\overline{\overline{C}} - 2\ \mathrm{tr}\left(\overline{\overline{C}}\right)\overline{\overline{A}} - 3\left(\overline{\overline{C}}\cdot\overline{\overline{A}} + \overline{\overline{A}}\cdot\overline{\overline{C}}\right) + 6\overline{\overline{\mathbb{A}}}_4 : \overline{\overline{C}}\right] \tag{2.52}$$

Realizing the drawback of the theory presented in Eq. (2.52), Phelps and Tucker [79] formulated a relation for anisotropic diffusion in the form of Eq.(2.51) but they redefined the fiber mobility tensor as a function of the rate of deformation, $\overline{\overline{D}}$, the second moment of fiber orientation, $\overline{\overline{A}}$, and constants $b_1 - b_5$ being empirically fit parameters seen in Eq. (2.53):

$$\overline{\overline{C}} = b_1\overline{\overline{I}} + b_2\overline{\overline{A}} + b_3\overline{\overline{A}}^2 + \frac{b_4}{\dot{\gamma}}\overline{\overline{D}} + \frac{b_5}{\dot{\gamma}^2}\overline{\overline{D}}^2 \tag{2.53}$$

Phelps and Tucker [79] also combined the work of Wang et al. [76] to form an anisotropic form of the reduced strain closure model shown in Eq. (2.54):

$$\begin{aligned}\frac{D\overline{\overline{A}}}{Dt} = {} & \overline{\overline{W}}\cdot\overline{\overline{A}} - \overline{\overline{A}}\cdot\overline{\overline{W}} + \xi\left\{\overline{\overline{D}}\cdot\overline{\overline{A}} + \overline{\overline{A}}\cdot\overline{\overline{D}} - 2\left[\overline{\overline{\mathbb{A}}}_4 + (1-\kappa)\left(\overline{\overline{\mathbb{L}}}_4 - \overline{\overline{\mathbb{M}}}_4 : \overline{\overline{\mathbb{A}}}_4\right) : \overline{\overline{D}}\right]\right\} \\ & + \dot{\gamma}\left\{2\left[\overline{\overline{C}} - (1-\kappa)\mathbb{M}_4 : \overline{\overline{C}}\right] - 2\kappa\ \mathrm{tr}\left(\overline{\overline{C}}\right)\overline{\overline{A}} - 5\left(\overline{\overline{C}}\cdot\overline{\overline{A}} + \overline{\overline{A}}\cdot\overline{\overline{C}}\right)\right. \\ & \left. + 10\left[\overline{\overline{A}} + (1-\kappa)\left(\overline{\overline{\mathbb{L}}}_4 - \overline{\overline{\mathbb{M}}}_4 : \overline{\overline{\mathbb{A}}}_4\right)\right] : \overline{\overline{C}}\right\}\end{aligned} \tag{2.54}$$

If the material is isotropic the fiber mobility tensor shown in Eq. (2.53) reduces down to the constant, C_I, and Eq. (2.41) is recovered. Parameter selection for the fiber mobility tensors proposed in Eq. (2.49) and Eq. (2.53) is very important and interested readers are directed to Koch [77] and Phelps and Tucker [79]. All of the modeling efforts including the anisotropic nature of the materials has not been applied to long glass fiber systems as of yet so comments on the accuracy of such modeling efforts cannot be made. Again, Eq. (2.54) maintains objectivity

but since the coordinate frame is not moving from an industrial modeling standpoint it is again unnecessary.

2.4.2 Flexible Fiber Orientation Models

Models for flexible fiber orientation may be necessary in the processing of composites with longer fibers. Forgacs and Mason [41, 80] were the first to quantitatively address fiber bending as an issue in studying flexible fiber dynamics. Hinch [81] addressed the issue of a single deformable thread in a viscous media as shown in Eq. (2.55) and Eq. (2.56) assuming laminar flow conditions:

$$\frac{\partial \mathbf{x}}{\partial t} = \nabla \mathbf{v} \cdot \mathbf{x} + \frac{\partial T}{\partial s}\frac{\partial \mathbf{x}}{\partial s} + \frac{1}{2}T\frac{\partial^2 \mathbf{x}}{\partial s^2} \tag{2.55}$$

$$\frac{\partial^2 T}{\partial s^2} - \frac{1}{2}\left(\frac{\partial^2 \mathbf{x}}{\partial s^2}\right)^2 T = -\frac{\partial \mathbf{x}}{\partial t}\cdot \mathbf{D} \cdot \frac{\partial \mathbf{x}}{\partial t} \tag{2.56}$$

In Eq. (2.55) and Eq. (2.56), s is the arc length of the fiber, $\mathbf{x}$ is a function of both s and t and represents a three-dimensional position vector of each point along the fiber, the tensor T is also and function of s and t and $\mathbf{v}$ is the velocity field of the fluid. In Hinch's theory a force balance on the hydrodynamic effects of the matrix is used to describe the motion of a single flexible thread in a dilute solution. The model consists of two coupled partial differential equations to describe the motion of the thread for configuration and tension, respectively. The solution of Eqs. (2.55) and (2.56) is not useful because it is only valid in dilute solutions.

Goddard and Huang [82] improved on this basic model for dilute suspensions by including fiber mobility in Eq. (2.55) and Eq. (2.56). This inclusion allowed for the model to take into account hydrodynamic effects of nearby fibers. The resulting equations are shown in Eq. (2.57) and Eq. (2.58) where $\mathbf{K}_T$ is the transverse mobility tensor and K_L and K_N are the lateral and normal components of the mobility tensor, respectively:

$$\frac{\partial \mathbf{x}}{\partial t} - K_L\frac{\partial T}{\partial s}\frac{\partial \mathbf{x}}{\partial s} = T\mathbf{K}_T \cdot \frac{\partial^2 \mathbf{x}}{\partial s^2} + \mathbf{v}(\mathbf{x},t) \tag{2.57}$$

$$\frac{\partial}{\partial s}\left(K_L\frac{\partial T}{\partial s}\right) - \left(\frac{\partial^2 \mathbf{x}}{\partial s^2}\right)\cdot\left(\frac{\partial^2 \mathbf{x}}{\partial s^2}\right)K_N T = -\frac{\partial \mathbf{x}}{\partial s}\cdot\frac{\partial \mathbf{v}(\mathbf{x},t)}{\partial s} \tag{2.58}$$

In researching this theory, few authors have utilized Eqs. (2.57) and (2.58) mainly because they fail to capture some of the dynamics, and they are cumbersome to solve.

Strautins and Latz [55] developed a continuum based model for semi flexible fiber systems through the use of moment orientation tensors. For fibers that can be considered semi-flexible, two vectors (labeled $\mathbf{p}$ and $\mathbf{q}$) are defined along the half-length of a deflected fiber seen in Figure 2.7 that are attached by a freely rotating joint with an internal resistivity to bending, k. If the fiber is strait, the internal spring is not loaded and there is no straitening force on the fiber.

If the fiber is in a deformed state (by a flow field acting upon it), the spring acts against the deformation to try and return the fiber to a strait state.

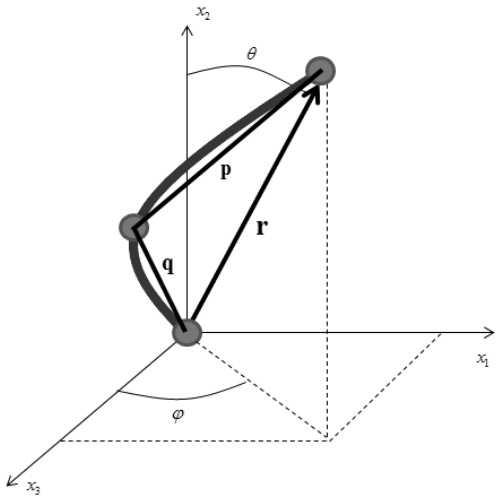

Figure 2.7: Semi-flexible representation of fiber. The fiber is represented by the vectors $\overline{p}$ and $\overline{q}$, an end-to-end vector $\overline{r}$, an internal resistivity to bending k, and the azimuthal and zenith angles j and q, respectfully.

Efficient modeling of the evolution of orientation in a complex flow field for semi-flexible fibers required the development of kinematic equations based on orientation tensors. In order to simplify the system so that it could be solved with relative ease, a number of assumptions were made. The first assumption was that the fibers are only semi-flexible and the bending occurs only at the center of the fiber so that $\overline{p} = -\overline{q}$. Also, it was assumed that at any time, the first and second moments of the orientation vectors $\overline{p}$ and $\overline{q}$ were equal. Three moment tensors arose in this analysis for the semi-flexible system, shown in Eqs., (2.59), and (2.61):

$$\overline{\overline{A}} = \iint \overline{p}\overline{p}\psi\left(\overline{p},\overline{q},t\right)d\overline{p}\,d\overline{q} \tag{2.59}$$

$$\overline{\overline{B}} = \iint \overline{p}\overline{q}\psi\left(\overline{p},\overline{q},t\right)d\overline{p}\,d\overline{q} \tag{2.60}$$

$$\overline{C} = \iint \overline{p}\psi\left(\overline{p},\overline{q},t\right)d\overline{p}\,d\overline{q} \tag{2.61}$$

The $\overline{\overline{A}}$ orientation tensor in Eq. (2.59) represents the second moment of the orientation distribution function and is similar to the moment tensor for rigid fiber theory. In the theory for semi-flexible fibers a $\overline{\overline{B}}$ tensor can also be defined which is a mixed second moment of the orientation distribution function given by Eq. (2.60). Finally, the first moment of the orientation distribution is given by the $\overline{C}$ tensor in Eq. (2.61) and is not zero for a semi-flexible fiber (unlike the rigid fiber model where the odd numbered moments are always zero). In the limit where the semi-flexible fiber is in a straitened state (there are no internal forces trying to straighten the fiber) then the mixed second moment (Eq.(2.60)) and the first moment (Eq.(2.61)) go to zero and

only the second moment of the orientation distribution function is non-zero, recovering a rigid fiber model.

Similarly, the end-to-end tensor, $\overline{\overline{r}}$, can also be integrated over orientation space to form an end-to-end moment tensor seen in Eq. (2.62) which represents the average orientation of a slightly bent fiber:

$$\overline{\overline{r}} = \iint l_b^2 (\overline{p} - \overline{q})(\overline{p} - \overline{q}) \psi (\overline{p}, \overline{q}, t) d\overline{p}\, d\overline{q} \tag{2.62}$$

Strautins and Latz [55] defined equations to describe the orientation kinetics of non-Brownian semi-flexible fibers in dilute solution given in Eqs. (2.63) to (2.66) where l_b is the length of the fiber, k is a resistivity to bending, and $\overline{m}$ is a vector quantity that takes into account fiber bending due to the flow field.

$$\begin{aligned} \frac{D\overline{\overline{A}}}{Dt} &= \overline{\overline{W}} \cdot \overline{\overline{A}} - \overline{\overline{A}} \cdot \overline{\overline{W}} + \xi \left(\overline{\overline{D}} \cdot \overline{\overline{A}} + \overline{\overline{A}} \cdot \overline{\overline{D}} - 2\overline{\overline{D}} : \overline{\overline{\mathbb{A}}}_4 \right) \ldots \\ &+ \frac{l_b}{2} \left[\overline{C}\overline{m} + \overline{m}\overline{C} - 2\left(\overline{m} \cdot \overline{C} \right) \overline{\overline{A}} \right] + 2k \left(\overline{\overline{B}} - \overline{\overline{A}}\ \mathrm{tr}\left(\overline{\overline{B}} \right) \right) \end{aligned} \tag{2.63}$$

$$\begin{aligned} \frac{D\overline{\overline{B}}}{Dt} &= \overline{\overline{W}} \cdot \overline{\overline{B}} - \overline{\overline{B}} \cdot \overline{\overline{W}} + \xi \left(\overline{\overline{D}} \cdot \overline{\overline{B}} + \overline{\overline{B}} \cdot \overline{\overline{D}} - \left(2\overline{\overline{D}} : \overline{\overline{A}} \right) \overline{\overline{B}} \right) \ldots \\ &+ \frac{l_b}{2} \left[\overline{C}\overline{m} + \overline{m}\overline{C} - 2\left(\overline{m} \cdot \overline{C} \right) \overline{\overline{B}} \right] + 2k \left(\overline{\overline{A}} - \overline{\overline{B}}\ \mathrm{tr}\left(\overline{\overline{B}} \right) \right) \end{aligned} \tag{2.64}$$

$$\frac{D\overline{C}}{Dt} = \nabla \overline{v}^T \cdot \overline{C} - \left(\overline{\overline{A}} : \nabla \overline{v}^T \right) \overline{C} + \frac{l_b}{2} \left[\overline{m} - \overline{C} \left(\overline{m} \cdot \overline{C} \right) \right] - k\overline{C} \left[1 - \mathrm{tr}\left(\overline{\overline{B}} \right) \right] \tag{2.65}$$

$$\overline{m} = \sum_{i=1}^{3} \sum_{j=1}^{3} \sum_{k=1}^{3} \frac{\partial^2 v_i}{\partial x_j \partial x_k} A_{jk} \overline{\delta}_i \tag{2.66}$$

While the semi-flexible fiber system described above yield good results for dilute and semi-dilute systems, Strautins and Latz [55] were originally concerned with modeling only the bending that occurs in long fiber systems and disregarded any fiber-fiber interaction.

In concentrated suspensions fiber interaction is omnipresent and must be accounted for. Ortman et al. [38] adapted Eqs. (2.63) to (2.66) for concentrated systems by incorporating the slip parameter, α, and the isotropic rotary diffusion term given in Eqs. (2.67) to (2.70):

$$\begin{aligned} \frac{D\overline{\overline{A}}}{Dt} &= \alpha \left[\overline{\overline{W}} \cdot \overline{\overline{A}} - \overline{\overline{A}} \cdot \overline{\overline{W}} + \xi \left(\overline{\overline{D}} \cdot \overline{\overline{A}} + \overline{\overline{A}} \cdot \overline{\overline{D}} - 2\overline{\overline{D}} : \overline{\overline{\mathbb{A}}}_4 \right) \ldots \right. \\ &\left. + \frac{l_b}{2} \left[\overline{C}\overline{m} + \overline{m}\overline{C} - 2\left(\overline{m} \cdot \overline{C} \right) \overline{\overline{A}} \right] + 2k \left(\overline{\overline{B}} - \overline{\overline{A}}\ \mathrm{tr}\left(\overline{\overline{B}} \right) \right) - 6C_I \dot{\gamma} \left(\overline{\overline{A}} - \tfrac{1}{3} \overline{\overline{I}} \right) \right] \end{aligned} \tag{2.67}$$

$$\frac{D\bar{\bar{B}}}{Dt}=\alpha\Big[\bar{\bar{W}}\cdot\bar{\bar{B}}-\bar{\bar{B}}\cdot\bar{\bar{W}}+\xi\Big(\bar{\bar{D}}\cdot\bar{\bar{B}}+\bar{\bar{B}}\cdot\bar{\bar{D}}-\Big(2\bar{\bar{D}}:\bar{\bar{A}}\Big)\bar{\bar{B}}\Big)\ldots$$
$$+\frac{l_b}{2}\Big[\bar{C}\bar{m}+\bar{m}\bar{C}-2\big(\bar{m}\cdot\bar{C}\big)\bar{\bar{B}}\Big]+2k\Big(\bar{\bar{A}}-\bar{\bar{B}}\ \mathrm{tr}\Big(\bar{\bar{B}}\Big)\Big)-4C_I\dot{\gamma}\bar{\bar{B}}\Big] \tag{2.68}$$

$$\frac{D\bar{\bar{C}}}{Dt}=\alpha\Big[\nabla\bar{v}^T\cdot\bar{\bar{C}}-\Big(\bar{\bar{\mathrm{A}}}:\nabla\bar{v}^T\Big)\bar{\bar{C}}+\frac{l_b}{2}\Big[\bar{m}-\bar{\bar{C}}\Big(\bar{m}\cdot\bar{\bar{C}}\Big)\Big]-k\bar{\bar{C}}\Big[1-\mathrm{tr}\Big(\bar{\bar{B}}\Big)\Big]-2C_I\dot{\gamma}\bar{\bar{C}}\Big] \tag{2.69}$$

$$\bar{m}=\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3}\frac{\partial^2 v_i}{\partial x_j\,\partial x_k}A_{jk}\bar{\delta}_i \tag{2.70}$$

Reasonable comparisons have been made with experimental data using the modified semi-flexible fiber equations in concentrated suspensions in both simple shear flows [56] and center-gated disks using the Hele-Shaw approximation [83].

2.4.3 Model Objectivity

Oldroyd first suggested that any constitutive equation (for fluids, fibers, etc.) should have three properties: (1) be independent of the frame of reference, (2) be independent of the position in space, the translational motion, and the rotational motion of the fluid element, and (3) be independent of the stress and strain in the neighboring fluid elements [84, 85]. From these requirements, a "convected" coordinate system is utilized whereby the coordinate system translates, rotates, and deforms with the fluid element [53]. Detailed examples of convected coordinates can be found in any introductory text to continuum mechanics [86-89].

In the modeling equations presented (§ 2.4.1 and § 2.4.2) only some equations maintain objectivity. Both the original model of Jeffery model (Eq.(2.40)) and the Folgar-Tucker model (Eq. (2.41)) maintain objectivity. This is because the co-rotational (Jaumann) derivative can still be formed by moving the terms containing vorticity to the left-hand side of the equation. This can be seen generally in Eq. (2.71) .

$$\frac{\mathfrak{D}\bar{\bar{\Lambda}}}{\mathfrak{D}t}=\frac{D\bar{\bar{\Lambda}}}{Dt}+\Big\{\bar{\bar{W}}\cdot\bar{\bar{\Lambda}}-\bar{\bar{\Lambda}}\cdot\bar{\bar{W}}\Big\} \tag{2.71}$$

Once the Folgar-Tucker model is multiplied by the slip parameter, α, Eq. (2.71) can no longer be formed by moving the vorticity terms to the left hand side. Similarly, when the Bead-Rod model is multiplied by the retardation factor, it too is no longer considered objective. The newer models presented in Eq. (2.45) and Eq. (2.54) does maintain objectivity by moving the retardation factor out of the hydrodynamic contributions.

The lack of objectivity may not cause a loss of accuracy in the numerical predictions of fiber orientation in industrial processes because the coordinate frame does not translate or rotate while the molding operation occurs. The models presented in Eq. (2.44) and Eqs. (2.67) to (2.70) are still valid to simulate fiber orientation but they violate basic continuum mechanics principles.

2.4.4 Closure Approximations

The choice of closure approximation for the fourth moment of the orientation tensor, $\mathbb{A}_4$, is not trivial and has been the subject of significant research [31]. The fourth moment of the orientation tensor is given by Eq. (2.31) which can be expanded to the matrix form seen in Eq. (2.72):

$$\overline{\overline{\mathbb{A}}}_4 = A_{ijlk} = \begin{bmatrix} a_{1111} & a_{1122} & a_{1133} & a_{1123} & a_{1113} & a_{1112} \\ a_{1122} & a_{2222} & a_{2233} & a_{2223} & a_{2213} & a_{2212} \\ a_{1133} & a_{2233} & a_{3333} & a_{3323} & a_{3313} & a_{3312} \\ a_{1123} & a_{2223} & a_{3323} & a_{2323} & a_{2313} & a_{2313} \\ a_{1113} & a_{2213} & a_{3313} & a_{2313} & a_{1313} & a_{1312} \\ a_{1112} & a_{2212} & a_{3312} & a_{2312} & a_{1312} & a_{1212} \end{bmatrix} \tag{2.72}$$

Due to symmetries in the matrix in Eq. (2.72) only 15 independent components remain. The linear closure by Hand [90] was the first use of a closure approximation to approximate the higher order moments an is shown in Eq. (2.73).

$$\begin{aligned} A_{ijkl} \approx & -\frac{1}{35}\left(\delta_{ij}\delta_{kl} + \delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right)... \\ & + \frac{1}{7}\left(A_{ij}\delta_{kl} + A_{ik}\delta_{jl} + A_{il}\delta_{jk} + A_{kl}\delta_{ij} + A_{jl}\delta_{ik} + A_{jk}\delta_{il}\right) \end{aligned} \tag{2.73}$$

One of the most popular closure approximations to date was first proposed by Doi [91] in 1981 and is known as the quadratic closure approximation shown in Eq. (2.74):

$$A_{ijkl} \approx A_{ij}A_{kl} \tag{2.74}$$

The quadratic closure approximation is exact for highly aligned systems or for systems undergoing pure extensional flow but is used widely because of the simple nature and computational efficiency. A combination of the linear and quadratic closures, called the hybrid closure approximation, was first reported by Advani and Tucker [62]. The hybrid closure has been used to some extent in processing calculations for part warpage [92] and even 3-dimensional short fiber orientation predictions for center-gated molds [93]. Even though these closures offer simplicity, they do not obey the required of characteristic of symmetry $\left(A_{ijkl} \neq A_{ikjl} \ \forall \ i,j,k,l\right)$ shown in Eq. (2.72).

Verleye and Dupret [94] introduced an objective closure approximation that maintained the condition of symmetry known as the "natural" closure approximation. The closure approximation is of the form shown in Eq. (2.75) and Eq. (2.76) and is the basis for many of the currently accepted forms of orthotropic closure approximations:

$$A_{ijkl} = \beta_1 S\left(\delta_{ij}\delta_{kl}\right) + \beta_2 S\left(\delta_{ij}A_{kl}\right) + \beta_3 S\left(A_{ij}A_{kl}\right) + \beta_4 S\left(\delta_{ij}A_{km}A_{ml}\right) + \beta_5 S\left(A_{ij}A_{km}A_{ml}\right) + \beta_6 S\left(A_{im}A_{mj}A_{kn}A_{nl}\right) \tag{2.75}$$

where S is the operator given in Eq. (2.76):

$$S\left(T_{ijkl}\right) = \frac{1}{24}\left(T_{ijkl} + T_{jikl} + T_{jkil} + \text{..... 24 terms}\right) \tag{2.76}$$

Cintra and Tucker [95] published a comprehensive work on orthotropic smooth closures and the necessary conditions for a closure to be symmetric while maintaining objectivity (§ 2.4.3). Other researchers [96, 97] explored the idea of eigenvalue based closures which assumes that the principle directions of the second-order moment orientation tensor define the planes of the fourth-order moment orientation tensor.

Invariant based fitting closures are of particular interest because the definition of the constants $\beta_1, \beta_2, \ldots$ in Eq. (2.75) are functions of the invariants of the orientation tensor, $\overline{\overline{A}}$ [98, 99]. The invariant based orthotropic fitted (IBOF) closure that is widely used was introduced by Chung and Kwon [100] and shows accurate predictions of fiber orientation over a wide range of flow conditions and fiber interaction coefficients. For the invariants, the frame of reference is taken as the principle frame of the second-order moment orientation tensor so that all terms vanish except for the diagonal components, A_{ii}. From this frame of reference, all non-diagonal terms go to zero in both the second- and fourth-order moment tensors so that the closure remains orthotropic. An extensive discussion of the validity of the IBOF closure can be found in Jack and Smith [101]. Sixth-order closure approximations have recently been studied by Jack and Smith [102, 103] because of sixth-order moments that have arisen in certain equations.

The use of the IBOF closure has been shown to work well in more complex flow fields and is thus the closure approximation of choice for the simulations in this work involving the 4th movement of the orientation distribution function [100, 104].

2.4.5 Effect of Brownian Motion

Brownian motion refers to the random movement of particles suspended in a fluid. The movement of the particle is a result of liquid molecules randomly impacting the particle in the fluid. For systems in which Brownian motion is a substantial effect the particle of interest must be small enough to be effected by these random motions. Brownian motion can be translational (where the position of the particle, molecule, etc. is changed) or rotational (where the unit normal of the surface direction is changed). The effect of Brownian motion on the suspensions is directly correlated to the magnitude of the Péclet number, given in Eq. (2.77) where D_r is the rotary diffusivity of the particle and $\dot{\gamma}$ is the magnitude of the rate of strain [105].

$$\text{Pe} = \frac{\dot{\gamma}}{D_r} \tag{2.77}$$

When $Pe \leq 1$, Brownian motion must be taken into account but when $Pe \gg 1$ Brownian motion is negligible. For a population of fibers with a length greater than $10\mu m$ Brownian motion can be neglected [37]. Hence, for the fibers under consideration here Brownian motion is neglected.

2.4.6 Summary of Fiber Orientation Equations

Predicting the orientation of fibers in suspension has been the subject of a significant amount of research starting with Jeffery's [63] equation for prolate spheroids in 1922. From that work a number of studies were launched on understanding how rigid bodies (spheroids, rods, etc.) in dilute media orient in different flow fields. As the need for concentrated suspensions in industrial applications grew, so did the attempts at modeling them with the development of the Folgar-Tucker model and the like. At the present time, a number of models exist to predict both rigid and semi-flexible concentrated fiber systems. The drawback of these models is that many of the terms which have been added to account for the higher concentration of fibers have been done so to decrease the discrepancy between predicted values and experimental values of orientation. As a consequence, there are phenomenological constants that must be fitted to the experimental data so that the predictions of fiber orientation are based on experimental data at some junction in simulation.

2.5 Numerical Predictions of Fiber Orientation

Injection molding is a frequently used method of forming thermoplastic parts reinforced with glass fibers [106]. The ease at which the process can be automated makes injection molding an attractive choice when high volume production is desired such as in the manufacture of automotive parts.

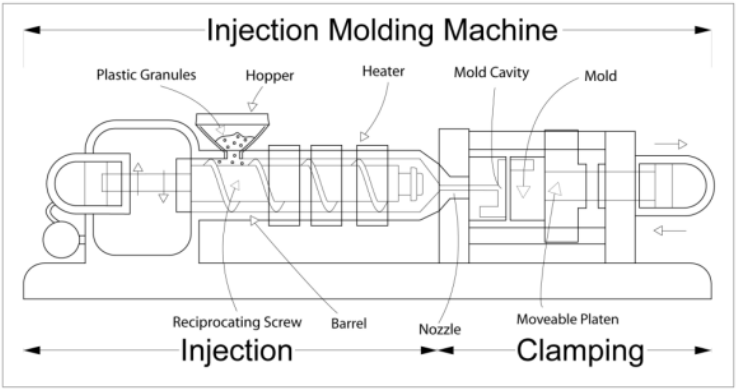

Figure 2.8: Diagram of a typical injection molding machine. Two main regions are described involving the injection and clamping sections of the machine. Figure used under the Creative Commons Attribution 3.0 License.

The injection portion of the machine, shown in Figure 2.8, is an extruder whereby the raw pellets are fed through a hopper into a heated barrel with a rotating screw. Heat is also generated by the viscous shearing of the pellets against the barrel. The pellets are melted in the barrel and the screw acts as a piston, pushing the liquid resin into the mold. The molten polymer is then allowed to cool in the mold cavity and the part is ejected.

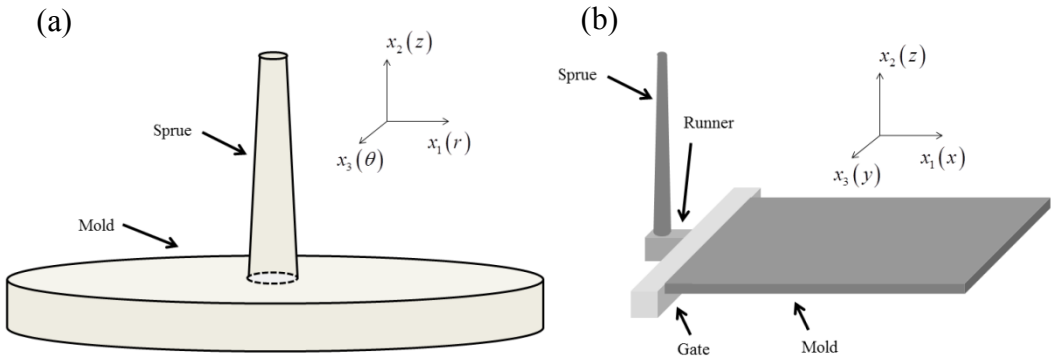

Figure 2.9: Diagram of typical geometries used in fiber orientation simulation in (a) center-gated disk and (b) end-gated plaque.

For the purpose of this work, the numerical simulations and modeling will be confined to the fluid flowing into the mold cavity which is where the fiber orientation predictions are generated. Two simple molding geometries used in a number of simulations are the center-gated disk and end-gated plaque, shown in (a) and (b) of Figure 2.9, respectfully. In the center-gated disk, the fluid enters at the top of the sprue, travels down the sprue, and enters the mold cavity. A number of authors have taken advantage of the axisymmetry of the flow field in the cavity to simplify their simulations [76, 83, 107]. In the end-gated mold, fluid again travels down the sprue, through the runner, and then into the gate and mold areas. The end-gated system has not been studied as much as the center-gated system because of the fully three-dimensional nature of the end-gated mold (compared to the 2.5 dimensional center-gated disk).

The amount of work that has been done in the area of flow fields without the addition of fillers in complex geometries is vast and a comprehensive review of literature has been the subject of entire texts [106]. However, it is important to point out specific advancements that have been made in the area of computational fluid dynamics when dealing with fiber suspensions. First, the effect of a non-Newtonian suspending matrix is discussed in § 2.5.1. Next, a description of simulating systems by either a Hele-Shaw method or full simulation method is presented in § 2.5.2. A review of fiber orientation simulations involving center-gated and end-gated systems is given in § 2.5.3 and § 2.5.4, respectively.

2.5.1 Effect of a non-Newtonian Suspending Liquid

In many publications, to simplify the suspension being studied, a Newtonian fluid is employed to understand fiber dynamics [6, 57, 108-110]. This is a good first approximation for many fluids but a more realistic model would take into account non-Newtonian behavior. Commonly a Generalized Newtonian Fluid model (GNF) is used to represent the suspending medium presented in Eq. (2.78) where $\overline{\overline{\tau}}$ is the extra stress tensor, $\eta(\dot{\gamma})$ means that the viscosity is a function of the shear rate and $\overline{\overline{\dot{\gamma}}}$ is the rate of strain tensor.

$$\overline{\overline{\tau}} = \eta(\dot{\gamma})\overline{\overline{\dot{\gamma}}} \tag{2.78}$$

A number of empiricisms for $\eta(\dot{\gamma})$ in GNF models have been successfully used in describing neat matrices including, but not limited to, the Power-Law model, Carreau model, and for a number of temperature dependent situations, the Cross model [111].

In certain systems, such as in center-gated disks and end-gated plaques, shear-rates can span two orders of magnitude. A number of models can be used to represent this behavior but one model of particular interest is the Carreau-Yassuda model, shown in Eq. (2.79), which has the ability to capture the viscosity at high and low shear rates. In Eq. (2.79) η_0 is the zero shear viscosity, λ governs the onset of shear thinning and n is a parameter governing the degree of shear-thinning of the matrix.

$$\eta_m(\dot{\gamma}) = \eta_0\left[1+(\lambda\dot{\gamma})^a\right]^{\frac{n-1}{a}} \tag{2.79}$$

A qualitative viscosity curve of what the Carreau-Yasuda model predicts is shown in Figure 2.10 showing the ability to capture both Newtonian and non-Newtonian effects.

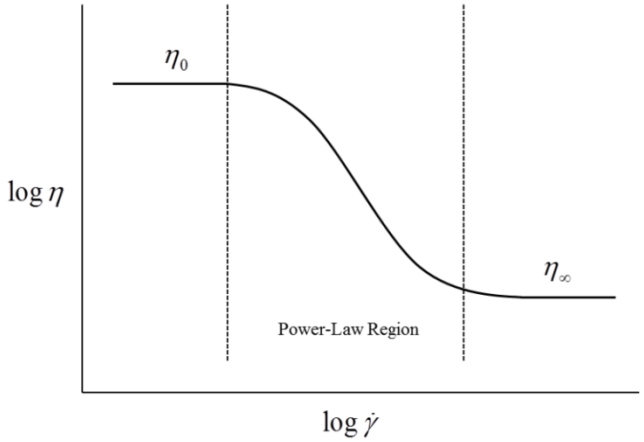

Figure 2.10: Qualitative depiction of Carreau-Yasuda model showing the ability to capture the Newtonian and non-Newtonian behavior of shear-thinning fluids. Log-log plot of viscosity vs. shear rate where h_0 and $h_\bullet$ are the zero-shear viscosity and infinite shear viscosity, respectively.

2.5.2 Hele-Shaw Flow vs. Full Simulation Effects

Simulating fiber orientation in molding geometries typically requires a number of assumptions to be made about the system in question. One significant assumption about the flow field in the mold cavity is the implementation of Hele-Shaw flow which disregards a number of important features about the flow field [83, 112, 113]. Only few simulations have recognized the importance of including such flow features in fiber orientation simulations in subsequent publications [114, 115].

2.5.2.1 Hele-Shaw Formulation

Many early simulations of fiber orientation employed the Hele-Shaw flow approximation which uses lubrication theory in conjunction with laminar flow assumption to yield analytical solutions to complex flow problems [111, 116].

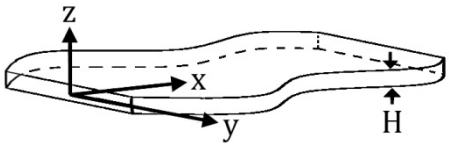

Figure 2.11: Qualitative representation of Hele-Shaw geometry where the gradient in one direction (z-direction) clearly dominates terms on the momentum equation.

From the lubrication approximation it is appropriate to solve the problem in the limit of the cavity thickness approaching zero, or $H \to 0$. Also, the no-slip boundary $(\overline{v} = 0)$ condition is imposed at the walls. The Hele-Shaw formation is given for injection molding by Hieber and Shen [117] and is the current method of choice for most simulation packages [118, 119]. The pressure equations is given as Eq. (2.80):

$$\frac{\partial}{\partial x}\left(S\frac{\partial P}{\partial x} \right)+\frac{\partial}{\partial y}\left(S\frac{\partial P}{\partial y} \right)=0 \quad \text{where } S=\int_0^b \frac{z^2}{\eta}dz \tag{2.80}$$

The velocities can then be rewritten in terms of the pressure gradient to form the two relations shown in Eq. (2.81) where v_x is the velocity in the x-direction, v_y is the velocity in the y-direction, $\overline{z}$ is the normalized thickness and η is the viscosity:

$$v_x = -\frac{\partial P}{\partial x}\int_z^b \frac{\overline{z}}{\eta}d\overline{z}, \;\; v_y = -\frac{\partial P}{\partial y}\int_z^b \frac{\overline{z}}{\eta}d\overline{z} \tag{2.81}$$

Heiber and Shen [117] detailed an approach where in plane calculations were performed using a finite-element scheme and the out of plane calculations were performed via a finite difference method.

Hele-Shaw flow, while convenient and computationally efficient, assumes a velocity profile and initial fiber orientation at the inlet to the mold and disregards the "fountain flow" effect which occurs at the polymer-air boundary in the mold filling process. The two methods are compared in Figure 2.12 showing markedly different predictions for fiber orientation near the wall for 30 wt. % short glass fiber at 40% fill of a center-gated disk [70]. The lack of accurate modeling at the polymer-air interface led to a modified version of Hele-Shaw approximation introduced by Bay and Tucker [107] and Han and Im [114] for use in concentrated fiber suspensions. The modified Hele-Shaw method more accurately predicts the data reported by Bay and Tucker [70] by using an approximation technique for accounting for fountain flow and is an improvement on simulating fiber orientation in molding operations. To date, a number of simulation types use some form of Hele-Shaw flow to model fluid behind the moving front,

including an Eulerian (Fixed) mesh with a control volume approach [117, 120, 121], remeshing the frontal flow portion of the domain [122], or sometimes remeshing the domain entirely [123].

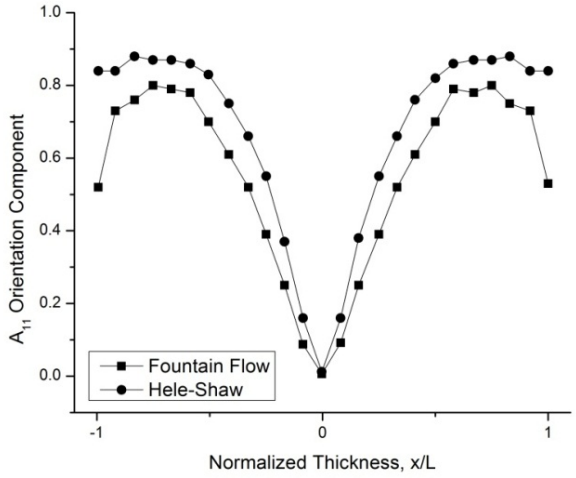

Figure 2.12: Comparison of Hele-Shaw approximation and the "fountain flow" effect in a center-gated disk at 40% of the fill domain for 30 wt. percent short glass fibers. When taking into account the fountain flow of mold cavity filling, theoretical predictions show much lower orientation predictions at the wall, which is commonly observed [70].

2.5.2.2 Fountain Flow

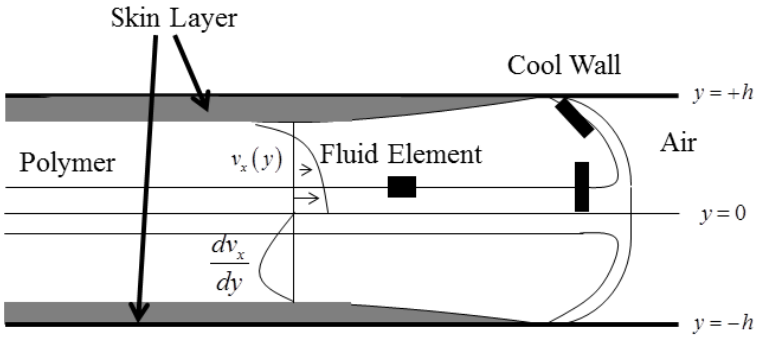

Figure 2.13: Qualitative figure of fountain flow effect. Adapted from Baird and Collias [124]. The fluid element reaches a stagnation point where the fluid is stretched and moved towards the outside wall.

Fountain flow, as it pertains to polymeric fluid filling a cavity, was introduced in the literature by Rose [125] and described the flow front near an advancing interface where one fluid is replacing another fluid in a two-phase problem. Fountain flow can be described by the fluid element in Figure 2.13 where a fluid element is moved towards the front of the flow and is subsequently stretched and moved towards the wall. A number of authors have studied fountain flow of a neat matrix including Tadmor [126], Gogos and Huang [127], Coyle et al. [128], and Mavridis et al. [129]. For as much work that has been done, modeling the filling of a cavity with a neat matrix is still a very complex problem.

Givler et al. [130] simulated fiber orientation for dilute suspensions including the fountain flow effect between two parallel plates utilizing FEM simulations. Alexandrou and Ahmed [131] studied the effects of fountain flow of a dilute suspension between parallel plates in an unsteady flow field. Vélez-García et al. [83] observed that the addition of the fountain flow at the flow front improved the agreement between predictions of fiber orientation and experimental data.

2.5.3 Fiber Orientation Simulations in a Center-Gated Disk

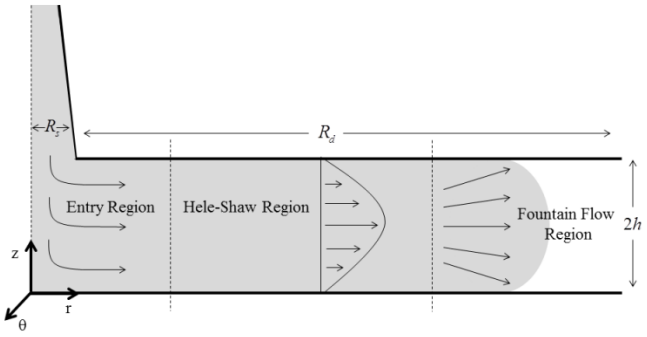

Figure 2.14: Qualitative representation of fluid filling a mold cavity with distinguishing features: Entry region where the fluid entering the mold is very near the gate, the Hele-Shaw region where lubrication approximation may be enforced, and the fountain flow region.

The flow fields within a center-gated disk make the geometry particularly useful in studying fiber orientation in mold filling. The three flow regions shown in Figure 2.14 were identified by Bay and Tucker [107] during the filling of a center-gated disk with a concentrated rigid fiber suspension: the gate region, the lubrication region, and the fountain flow region. Each of these regions contains some combination of shearing and extensional deformations. In the gate region, extension in the $r\theta$-plane governs the fiber orientation. In the lubrication region (Hele-Shaw region), small gap heights cause a domination of shearing deformation to occur and highly dictate fiber orientation. The flow front is a complicated region which was covered previously (See § 2.5.2).

2.5.3.1 Short Fiber Orientation Simulation for Hele-Shaw Flow in a Center-Gated Disk

Bay and Tucker [107] assumed that the majority of the mold cavity can be considered within the lubrication region (described in § 2.5.2.1) and thus simplifies the equations to solve for rigid fiber orientation. For the simulation of isothermal center-gated systems, the equations of motion are reduced significantly when the Hele-Shaw approach with a Newtonian fluid is taken. The velocity and pressure fields can be solved for directly yielding the analytical solution shown in Eq. (2.82) and Eq. (2.83) where cylindrical coordinates have been imposed:

$$v_r = \frac{1}{r}\frac{3Q}{8\pi H}\left(1-\frac{z^2}{H^2}\right) \tag{2.82}$$

$$v_z = \frac{v_r}{r} \tag{2.83}$$

From this solution fiber orientation can be solved for by decoupling the orientation and stress equations as done in Bay and Tucker [107] who showed good qualitative agreement with experimental data far from the gate and melt front [70, 132]. Chiang et al. [120, 133] used the Hele-Shaw flow approximation in conjunction with a finite element method (FEM) and finite difference method (FDM) to solve for the pressure and temperature fields.

Using only the Hele-Shaw region in the simulation of rigid fiber orientation does involve making broad assumptions about systems that may not generally apply. The Hele-Shaw assumption neglects the details of the flow field at the melt front (usually assumes a flat profile through the thickness) or abrupt changes in cavity thickness. Additionally, the Hele-Shaw approximation requires an assumed initial fiber orientation entering the mold cavity. A popular assumption for fiber orientation at the inlet of the mold is random $\left(\overline{\overline{A}} = \frac{1}{3}\overline{\overline{I}}\right)$ [107, 114].

2.5.3.2 Short Fiber Orientation Simulation Incorporating Entry and Fountain Flow Effect in a Center-Gated Disk

The limitations of the Hele-Shaw approximation led to Dupret and Vandershuren [134] and Bay and Tucker [107] to attempt to predict the velocity, temperature, and orientation fields in the fountain flow region using the method of Coyle et al. [128]. The full solution of flow and orientation equations in a steady-state radially diverging flow is given by Ranganathan and Advani [135] where the effect of particle number was studied. Ko and Young [124] investigated the fiber orientation in the thickness plane during injection molding while tracking the flow front advancement. Vincent et al. [136] used a decoupled approach and moving mesh technique to solve for short fiber orientation in a center-gated disk. VerWeyst and Tucker [137] studied the effect of entry flow for short glass fibers using a fully three-dimensional finite element technique. Velez-Garcia et al. [83] used a decoupled approach to make short glass fiber orientation predictions and found good agreement with experimental data. Ultimately, the choice of fiber orientation model and assumptions about the gate region play a far greater role in final orientation predictions so that the decoupled approach is a valid first approximation for calculating fiber orientation in the center-gated disk. Chung and Kwon [104] performed work studying the coupling effect in the advancing front and found that coupling effect was crucial especially in regions with high extension and only in the Hele-Shaw region did the coupling effect play a small role.

The short fiber orientation profiles in Figure 2.15 show the three different input values for use at the entrance to the mold cavity. The bottom region of the gate cross section shows fibers that are less aligned with the flow direction. This is due to the stagnation and change of direction that the fluid undergoes when the sprue is taken into account. Chung and Kwon [104] suggested that the short glass fiber orientation entering the mold cavity is asymmetric and plays a role in dictating fiber orientation in the mold cavity. This has been confirmed experimentally by Velez-Garcia et al. [83], and it was shown that the choice of initial fiber orientation values persist well into the Hele-Shaw region. Park and Kwon [138] simulated the orientation of short glass fiber for a non-isothermal system in a center-gated disk and noted that the temperature of the fluid does have an impact on the fiber orientation distribution in the mold cavity.

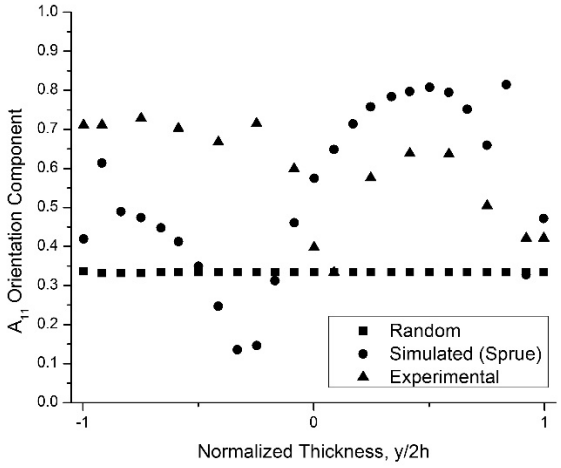

Figure 2.15: Comparison of different fiber orientation values in the gate region (0% fill) of the center-gated disk. (■) is random assumption through the thickness, (●) are the results of simulating short glass fiber orientation from the sprue to the gate [104], and (▲) are the experimental orientation values obtained for short glass fibers [83]. The data is for 30 wt. % short glass fiber in polypropylene matrix.

2.5.3.3 Long Fiber Orientation Simulations in a Center-Gated Disk

Up to this point, very little work has been done in modeling the orientation of long glass fiber composites in center-gated disks. Nguyen et al. [139] applied a short fiber model to a long glass fiber system injected into a center gated disk and saw good agreement between predictions and experimental orientation but the average fiber length of those systems was only around 1.55 mm. Ortman et al. [140] compared the ability of strain reduction Folgar-Tucker model (Eq. (2.44)) and the strain reduction Bead-Rod model (Eqs. (2.67) - (2.70)) to predict long glass fiber orientation (with an average length of 3.1 mm) in center-gated disks using Hele-Shaw flow assumption and experimentally obtained initial conditions at the gate and found good agreement with experimental data up to 40% of the mold fill.

2.5.4 Fiber Orientation Simulations in End-Gated Systems

The amount of literature for the calculation of fiber orientation in a three-dimensional channel is limited. Altan et al. [141] assumed a planar random inlet condition to a rectangular channel and computed the short fiber orientation using a coupled approach involving both the 4^{th} and 6^{th} order tensors of orientation. A number of other works have considered three-dimensional channel flow [107, 108, 141-143], sudden contractions [36, 144, 145], sudden expansions [143, 144, 146, 147], corner flows [148], and branching flows [149] for short glass fiber systems. At the time of this review, only one paper deals with the injection of long glass fibers into an end-gated geometry. Nguyen et al. [139] simulated the orientation of long glass fibers flowing into an end-gated mold using the RSC model (developed for short glass fiber systems) and found some agreement between experimental observations and predicted results along the center-line of the mold but did not compare results at other widths. The average fiber length reported by Nguyen et

al. [139] was ≈ 1.55 mm but a large population of fibers were significantly shorter, increasing the chance of a short glass fiber model accurately predicting fiber orientation.

2.5.5 Summary of Numerical Predictions of Fiber Orientation

From what has been presented in this section, it can be seen broad assumptions are often made when predicting the orientation of glass fibers in injection molding geometries. One major assumption, still widely used, is the Hele-Shaw assumption which neglects entry effects into the mold and also neglects the advancing front. Some simulations for short glass fiber have used an advancing front algorithm in conjunction with a Hele-Shaw flow assumption. While these hybrid numerical approaches predicted values closer to observed experimental short glass fiber orientation, an assumption still had to be made about fiber orientation at the inlet to the mold cavity. Recently, a method to simulate short glass fiber orientation was developed whereby the sprue and mold were simulated as a single domain which yielded good agreement with experimental fiber orientation values.

At the time of publishing this review, only one two works had reported on simulating long glass fibers in a center-gated disk. Both of the simulations utilized the Hele-Shaw method and did not account for either the advancing front or the entry effects into the mold. Ortman et al. [150] did account for the ability of a fiber to bend during the injection process with a semi-flexible fiber model which at least qualitatively represents long glass fibers systems more accurately. While Ortman et al. heavily simplified simulation was a good first step in predicting the orientation of semi-flexible fibers, it is necessary to extend a method where fewer assumptions are made to more accurately simulate the orientation kinetics of these complex systems.

Only one reference has reported on predicting semi-flexible fiber orientation in a fully three-dimensional system such as an end-gated plaque but used a short fiber model to predict a system composed most of short fibers. Predicting the orientation of long fibers in an end-gated plaque is the first step in understanding how semi-flexible fibers orient in three-dimensional flow fields. With this knowledge the solution method can be extended to geometrically complex parts.

2.6 References

[1] S. D. Bartus, U. K. Vaidya, and C. A. Ulven, "Design and development of a long fiber thermoplastic bus seat," *Journal of Thermoplastic Composite Materials,* vol. 19, pp. 131-154, Mar 2006.

[2] S. Toll and J. A. E. Manson, "Elastic Compression of a Fiber Network," *Journal of Applied Mechanics-Transactions of the Asme,* vol. 62, pp. 223-226, Mar 1995.

[3] M. Keshtkar, M. C. Heuzey, and P. J. Carreau, "Rheological behavior of fiber-filled model suspensions: Effect of fiber flexibility," *Journal of Rheology,* vol. 53, pp. 631-650, May-Jun 2009.

[4] M. Doi and S. F. Edwards, *The theory of polymer dynamics*. Oxford: Clarendon Press, 1986.

[5] S. M. Dinh and R. C. Armstrong, "A Rheological Equation of State for Semiconcentrated Fiber Suspensions," *Journal of Rheology,* vol. 28, pp. 207-227, 1984.

[6] D. G. Baird, A. P. R. Eberle, G. M. Velez-Garcia, and P. Wapperom, "Fiber orientation kinetics of a concentrated short glass fiber suspension in startup of simple shear flow," *Journal of Non-Newtonian Fluid Mechanics,* vol. 165, pp. 110-119, Feb 2010.

[7] D. G. Baird, A. P. R. Eberle, P. Wapperom, and G. M. Velez-Garcia, "Using transient shear rheology to determine material parameters in fiber suspension theory," *Journal of Rheology,* vol. 53, pp. 685-705, May-Jun 2009.

[8] D. G. Baird, A. P. R. Eberle, P. Wapperom, and G. M. Velez-Garcia, "Obtaining reliable transient rheological data on concentrated short fiber suspensions using a rotational rheometer," *Journal of Rheology,* vol. 53, pp. 1049-1068, Sep-Oct 2009.

[9] I. Taha and Y. F. Abdin, "Modeling of strength and stiffness of short randomly oriented glass fiber-polypropylene composites," *Journal of Composite Materials,* vol. 45, pp. 1805-1821, Aug 2011.

[10] P. J. Carreau, J. Thomasset, B. Sanschagrin, and G. Ausias, "Rheological properties of long glass fiber filled polypropylene," *Journal of Non-Newtonian Fluid Mechanics,* vol. 125, pp. 25-34, Jan 10 2005.

[11] L. H. Switzer and D. J. Klingenberg, "Flocculation in simulations of sheared fiber suspensions," *International Journal of Multiphase Flow,* vol. 30, pp. 67-87, Jan 2004.

[12] D. J. Klingenberg and L. H. Switzer, "Rheology of sheared flexible fiber suspensions via fiber-level simulations," *Journal of Rheology,* vol. 47, pp. 759-778, May-Jun 2003.

[13] M. W. Darlington and P. L. Mcginley, "Fiber Orientation Distribution in Short Fiber Reinforced-Plastics," *Journal of Materials Science,* vol. 10, pp. 906-910, 1975.

[14] F. Lisy, A. Hiltner, E. Baer, J. L. Katz, and A. Meunier, "Application of Scanning Acoustic Microscopy to Polymeric Materials," *Journal of Applied Polymer Science,* vol. 52, pp. 329-352, Apr 11 1994.

[15] K. M. Thomas, D.E., "Study of glass-fibre-reinforced thermoplastic mouldings," *Plastics and Rubber Processing and Applications,* pp. 99-108, 1976.

[16] A. Mavrich, F. Fondeur, H. Ishida, J. L. Koenig, and H. D. Wagner, "The Study of Fiber-Matrix Interactions Via Ft-Ir Microscopy and Nmr Imaging," *Journal of Adhesion,* vol. 46, pp. 91-102, 1994.

[17] A. Kriete, "Undesirable phenomena in 3D image ctometry," *Visualization Biology,* pp. 214-218, 1992.

[18] P. J. Hine, N. Davidson, R. A. Duckett, and I. M. Ward, "Measuring the Fiber Orientation and Modeling the Elastic Properties of Injection-Molded Long-Glass-Fiber-Reinforced Nylon," *Composites Science and Technology,* vol. 53, pp. 125-131, 1995.

[19] A. R. Clarke, G. Archenhold, and N. C. Davidson, "A Novel Technique for Determining the 3d Spatial-Distribution of Glass-Fibers in Polymer Composites," *Composites Science and Technology,* vol. 55, pp. 75-91, 1995.

[20] A. Clarke, N. Davidson, and G. Archenhold, "Measurements of Fiber Direction in Reinforced Polymer Composites," *Journal of Microscopy-Oxford,* vol. 171, pp. 69-79, Jul 1993.

[21] G. M. Velez-Garcia, P. Wapperom, V. Kunc, D. G. Baird, and A. Zink-Sharp, "Sample preparation and image aquisition using optical-reflective microscopy in the measurement of fiber orientation in thermoplastic composites," 2012.

[22] P. J. Hine and R. A. Duckett, "Fiber orientation structures and mechanical properties of injection molded short glass fiber reinforced ribbed plates," *Polymer Composites,* vol. 25, pp. 237-254, Jun 2004.

[23] I. Junkar, U. Cvelbar, A. Vesel, N. Hauptman, and M. Mozetic, "The Role of Crystallinity on Polymer Interaction with Oxygen Plasma," *Plasma Processes and Polymers,* vol. 6, pp. 667-675, Oct 14 2009.

[24] A. R. Clarke and C. N. Eberhardt. (2002). *Microscopy techniques for materials science*. Available: http://www.knovel.com/knovel2/Toc.jsp?BookID=846

[25] A. Einstein, "Eine neue Bestimmung der Molekuldimensionen," *Annalen der Physik,* vol. 19, pp. 289-306, 1906.

[26] M. Mooney, "The Viscosity of a Concentrated Suspension of Spherical Particles," *Journal of Colloid Science,* vol. 6, pp. 162-170, 1950.

[27] G. L. Hand, "A theory of dilute suspensions," *Archive for Rational Mechanics and Analysis,* vol. 7, pp. 81-86, 1961.

[28] H. Giesekus, "Strömungen mit konstantem Geschwindigkeitsgradienten und die Bewegung von darin suspendierten Teilchen.Teil I: Räumliche Strömungen," *Rheologica Acta,* vol. 2, pp. 101-112, 1962.

[29] H. Giesekus, "Strömungen mit konstantem Geschwindigkeitsgradienten und die Bewegung von darin suspendierten Teilchen.Teil II: Ebene Strömungen und eine experimentelle Anordnung zu ihrer Realisierung," *Rheologica Acta,* vol. 2, pp. 112-122, 1962.

[30] G. K. Batchelor, "The stress system in a suspension of force-free particles," *Journal of Fluid Mechanics,* vol. 41, pp. 545-570, 1970.

[31] D. H. Chung and T. H. Kwon, "Fiber orientation in the processing of polymer composites," *Korea-Australia Rheology Journal,* vol. 14, pp. 175-188, Dec 2002.

[32] G. K. Batchelor, "The stress generated in a non-dilute suspension of elongated particles by pure straining motion," *Journal of Fluid Mechanics,* vol. 46, pp. 813-829, 1971.

[33] E. S. G. Shaqfeh and G. H. Fredrickson, "The Hydrodynamic Stress in a Suspension of Rods," *Physics of Fluids a-Fluid Dynamics,* vol. 2, pp. 7-24, Jan 1990.

[34] J. L. Ericksen, "Transversely Isotropic Fluids," *Kolloid-Zeitschrift,* vol. 173, 1960.

[35] J. D. Goddard, "Tensile behavior of power-law fluids containing oriented slender fibers.," *Journal of Rheology,* vol. 22, pp. 615-622, 1978.

[36] G. G. Lipscomb, M. M. Denn, D. U. Hur, and D. V. Boger, "The Flow of Fiber Suspensions in Complex Geometries," *Journal of Non-Newtonian Fluid Mechanics,* vol. 26, pp. 297-325, Jan 1988.

[37] A. P. R. Eberle, D. G. Baird, P. Wapperom, and G. M. Velez-Garcia, "Obtaining reliable transient rheological data on concentrated short fiber suspensions using a rotational rheometer," *Journal of Rheology,* vol. 53, pp. 1049-1068, Sep-Oct 2009.

[38] K. C. Ortman, N. Agarwal, A. P. R. Eberle, D. G. Baird, P. Wapperom, and A. J. Giacomin, "Transient shear flow behavior of concentrated long glass fiber suspensions in a sliding plate rheometer," *Journal of Non-Newtonian Fluid Mechanics,* vol. 166, pp. 884-895, Sep 1 2011.

[39] A. Aitkadi and M. Grmela, "Modeling the Rheological Behavior of Fiber Suspensions in Viscoelastic Media," *Journal of Non-Newtonian Fluid Mechanics,* vol. 53, pp. 65-81, Jul 1994.

[40] J. Azaiez, "Constitutive equations for fiber suspensions in viscoelastic media," *Journal of Non-Newtonian Fluid Mechanics,* vol. 66, pp. 35-54, Sep 1996.

[41] O. L. Forgacs, Mason S.G., "Particle Motions in Sheared Suspensions IX. Spin and Deformation of Threadlike Particles," *Journal of Colloid Science,* vol. 14, pp. 457-472, 1959.

[42] S. Goto, H. Nagazono, and H. Kato, "The Flow Behavior of Fiber Suspensions in Newtonian Fluids and Polymer-Solutions .1. Mechanical-Properties," *Rheologica Acta,* vol. 25, pp. 119-129, Mar-Apr 1986.

[43] S. Yamamoto and T. Matsuoka, "A Method for Dynamic Simulation of Rigid and Flexible Fibers in a Flow Field," *Journal of Chemical Physics,* vol. 98, pp. 644-650, Jan 1 1993.

[44] S. Yamamoto and T. Matsuoka, "Dynamic simulation of microstructure and rheology of fiber suspensions," *Polymer Engineering and Science,* vol. 36, pp. 2396-2403, Oct 1996.

[45] R. F. Ross and D. J. Klingenberg, "Dynamic simulation of flexible fibers composed of linked rigid bodies," *Journal of Chemical Physics,* vol. 106, pp. 2949-2960, Feb 15 1997.

[46] C. F. Schmid, L. H. Switzer, and D. J. Klingenberg, "Simulations of fiber flocculation: Effects of fiber properties and interfiber friction," *Journal of Rheology,* vol. 44, pp. 781-809, Jul-Aug 2000.

[47] C. G. Joung, N. Phan-Thien, and X. J. Fan, "Direct simulation of flexible fibers," *Journal of Non-Newtonian Fluid Mechanics,* vol. 99, pp. 1-36, Apr 15 2001.

[48] G. Wang, W. Yu, and C. X. Zhou, "Optimization of the rod chain model to simulate the motions of a long flexible fiber in simple shear flows," *European Journal of Mechanics B-Fluids,* vol. 25, pp. 337-347, May-Jun 2006.

[49] M. Rajabian, C. Dubois, and M. Grmela, "Suspensions of semiflexible fibers in polymeric fluids: Rheology and thermodynamics," *Rheologica Acta,* vol. 44, pp. 521-535, Jul 2005.

[50] M. Keshtkar, M. C. Heuzey, P. J. Carreau, M. Rajabian, and C. Dubois, "Rheological properties and microstructural evolution of semi-flexible fiber suspensions under shear flow," *Journal of Rheology,* vol. 54, pp. 197-222, Mar-Apr 2010.

[51] M. Grmela and H. C. Ottinger, "Dynamics and thermodynamics of complex fluids. I. Development of a general formalism," *Physical Review E,* vol. 56, pp. 6620-6632, Dec 1997.

[52] H. C. Ottinger and M. Grmela, "Dynamics and thermodynamics of complex fluids. II. Illustrations of a general formalism," *Physical Review E,* vol. 56, pp. 6633-6655, Dec 1997.

[53] H. Hencky, "Ueber langsaeme stationare striimung in plastischen massen.," *Zeitschrift Fur Angewandte Mathematik Und Mechanik,* vol. 5, 1925.

[54] K. C. Ortman, "Assessing an Orientation Model and Stress Tensor for Semi-Flexible Glass Fibers in Polypropylene Using a Sliding Plate Rheometer: for the Use of Simulating Processes," Ph.D., Virginia Polytechnic and State University, 2011.

[55] U. Strautins and A. Latz, "Flow-driven orientation dynamics of semiflexible fiber systems," *Rheologica Acta,* vol. 46, pp. 1057-1064, Oct 2007.

[56] K. C. Ortman, P. Wapperom, A. Whittington, and D. G. Baird, "Using startup of steady shear flow in a sliding plate rheometer to determine material parameters for the purpose of predicting long fiber orientation," *Journal of Rheology,* vol. 56, pp. 955-981, 2012.

[57] J. Azaiez and R. Guenette, "Numerical modelling of the flow of fibre suspensions through a planar contraction," *Canadian Journal of Chemical Engineering,* vol. 80, pp. 1115-1125, Dec 2002.

[58] R. L. Schiek and E. S. G. Shaqfeh, "A Nonlocal Theory for Stress in Bound, Brownian Suspensions of Slender, Rigid Fibers," *Journal of Fluid Mechanics,* vol. 296, pp. 271-324, Aug 10 1995.

[59] D. A. Jack and D. E. Smith, "Assessing the use of tensor closure methods with orientation distribution reconstruction functions," *Journal of Composite Materials,* vol. 38, pp. 1851-1871, 2004.

[60] J. L. Ericksen, "Anisotropic Fluids," *Archive for Rational Mechanics and Analysis,* vol. 4, 1960.

[61] J. L. Ericksen, "Theory of Anisotropic Fluids," *Transactions of the Society of Rheology,* vol. 4, 1960.

[62] S. G. Advani and C. L. Tucker, "The Use of Tensors to Describe and Predict Fiber Orientation in Short Fiber Composites," *Journal of Rheology,* vol. 31, pp. 751-784, Nov 1987.

[63] G. B. Jeffery, "The Motion of Ellipsoidal Particles Immersed in a Viscous Fluid," *Proceedings of the Royal Society a-Mathematical Physical and Engineering Sciences,* vol. 100, pp. 161-179, 1922.

[64] B. J. Trevelyan, Mason, S.G., "Particle motions in sheared suspensions. I. Rotations," *Journal of Colloid Science,* vol. 6, pp. 354-367, 1951.

[65] S. E. Barbosa and M. G. Bibbo, "Fiber motion and rheology of suspensions with uniform fiber orientation," *Journal of Polymer Science Part B-Polymer Physics,* vol. 38, pp. 1788-1799, Jul 1 2000.

[66] S. G. Mason, "Particle Motions in Sheared Suspensions: Orientation and Interactions of Rigid Rods," *Proceedings of the Royal Society a-Mathematical Physical and Engineering Sciences,* vol. 238, 1956.

[67] S. G. M. R. St J Manley, "Particle motions in sheared suspensions II. Collisions of uniform spheres," *Journal of Colloid Science,* vol. 7, pp. 354-369, 1952.

[68] C. C. Ferguson, "Rotations of elongate rigid particles in slow non-Newtonian flows," *Tectonophysics,* vol. 60, pp. 247-262, 1979.

[69] F. Folgar and C. L. Tucker, "Orientation Behavior of Rigid Fibers in Concentrated Suspensions," *Journal of Rheology,* vol. 26, pp. 604-604, 1982.

[70] R. S. Bay and C. L. Tucker, "Fiber Orientation in Simple Injection Moldings .2. Experimental Results," *Polymer Composites,* vol. 13, pp. 332-341, Aug 1992.

[71] R. S. Bay, "Fiber orientation in injection molded composites: a comparison of theory and experiment," Ph.D. , University of Illinois, Urbana-Champaign, 1991.

[72] S. Ranganathan and S. G. Advani, "Fiber-Fiber Interactions in Homogeneous Flows of Nondilute Suspensions," *Journal of Rheology,* vol. 35, pp. 1499-1522, Nov 1991.

[73] N. Phan-Thien, X. J. Fan, R. I. Tanner, and R. Zheng, "Folgar-Tucker constant for a fibre suspension in a Newtonian fluid," *Journal of Non-Newtonian Fluid Mechanics,* vol. 103, pp. 251-260, Mar 25 2002.

[74] M. Sepehr, G. Ausias, and P. J. Carreau, "Rheological properties of short fiber filled polypropylene in transient shear flow," *Journal of Non-Newtonian Fluid Mechanics,* vol. 123, pp. 19-32, Oct 15 2004.

[75] R. I. Tanner, *Engineering Rheology*: Oxford University Press, 2000.

[76] C. L. Tucker, J. Wang, and J. F. O'Gara, "An objective model for slow orientation kinetics in concentrated fiber suspensions: Theory and rheological evidence," *Journal of Rheology,* vol. 52, pp. 1179-1200, Sep-Oct 2008.

[77] D. L. Koch, "A Model for Orientational Diffusion in Fiber Suspensions," *Physics of Fluids,* vol. 7, pp. 2086-2088, Aug 1995.

[78] X. J. Fan, N. Phan-Thien, and R. Zheng, "A direct simulation of fibre suspensions," *Journal of Non-Newtonian Fluid Mechanics,* vol. 74, pp. 113-135, Jan 1998.

[79] C. L. Tucker and J. H. Phelps, "An anisotropic rotary diffusion model for fiber orientation in short- and long-fiber thermoplastics," *Journal of Non-Newtonian Fluid Mechanics,* vol. 156, pp. 165-176, Feb 2009.

[80] O. L. Forgacs, Mason S.G., "Particle Motions in Sheared Suspensions X. Orbits of Flexible Threadlike Particles," *Journal of Colloid and Science,* vol. 14, pp. 473-491, 1959.

[81] E. J. Hinch, "The distortion of a flexible inextensible thread in a shearing flow," *Journal of Fluid Mechanics,* vol. 74, pp. 317-333, 1975.

[82] J. D. Goddard and Y. H. Huang, "On the Motion of Flexible Threads in a Stokes Shear Field," *Journal of Non-Newtonian Fluid Mechanics,* vol. 13, pp. 47-62, 1983.

[83] G. M. Velez-Garcia, S. M. Mazahir, P. Wapperom, and D. G. Baird, "Simulation of Injection Molding Using a Model with Delayed Fiber Orientation," *International Polymer Processing,* vol. 26, pp. 331-339, Jul 2011.

[84] R. B. Bird, *Dynamics of polymeric liquids*. New York: Wiley, 1977.

[85] J. G. Oldroyd, "An Approach to Non-Newtonian Fluid-Mechanics," *Journal of Non-Newtonian Fluid Mechanics,* vol. 14, pp. 9-46, 1984.

[86] W. Prager, *Introduction to Mechanics of Continua*. Boston, MA: Ginn, 1961.

[87] L. E. Malvern, *Introduction to the Mechanics of a Continuous Medium*. Englewood Cliffs, NJ: Prentice-Hall, 1969.

[88] A. S. Lodge, *Elastic Liquids*. New York, NY: Academic Press, 1964.

[89] R. B. Bird, R. C. Armstrong, and O. Hassager, *Dynamics of polymeric liquids. - 1 : Fluid mechanics*, 2.ed. ed. New York,N.Y.: Wiley, 1987.

[90] G. L. Hand, "A theory of anisotropic fluids," *Journal of Fluid Mechanics,* vol. 13, pp. 33-46, 1961.

[91] M. Doi, "Molecular-Dynamics and Rheological Properties of Concentrated-Solutions of Rodlike Polymers in Isotropic and Liquid-Crystalline Phases," *Journal of Polymer Science Part B-Polymer Physics,* vol. 19, pp. 229-243, 1981.

[92] R. Haag, "Optimization of the warpage of fiber reinforced thermoplastics by influencing fiber orientation," *Proceedings of the 61st ANTEC conference and exhibition,* vol. 49, pp. 661-664, 2003.

[93] W.-H. Yang, Hsu, D.C., Yang, V., "Computer simulation of 3D short fiber orientation in injection molding," *Proceedings of the 61st ANTEC conference and exhibition,* vol. 49.

[94] V. Verleye, Dupret, F., "Prediction of the fiber orientation in complex injection molded parts," *Proceedings of the ASME 1993 Winter Anuual Meeting,* vol. 175, pp. 139-163, 1993.

[95] J. S. Cintra and C. L. Tucker, "Orthotropic Closure Approximations for Flow-Induced Fiber Orientation," *Journal of Rheology,* vol. 39, pp. 1095-1122, Nov-Dec 1995.

[96] B. E. VerWeyst, C. L. Tucker, P. H. Foss, and J. F. O'Gara, "Fiber orientation in 3-D injection molded features - Prediction and experiment," *International Polymer Processing,* vol. 14, pp. 409-420, Dec 1999.

[97] D. H. Chung and T. H. Kwon, "Improved model of orthotropic closure approximation for flow induced fiber orientation," *Polymer Composites,* vol. 22, pp. 636-649, Oct 2001.

[98] F. Dupret, Verleye, V., "Modeling the flow of fiber suspensions in narrow gaps," *Advances in the flow and rheology of non-Newtonian fluids, Part B,* pp. 1347-1398, 1999.

[99] C. A. Petty, Parks, S.M., Shiwei, M.S., "Flow-induced alignment of fibers," *Proceedings of the ICCM-12 conference,* 1999.

[100] D. H. Chung and T. H. Kwon, "Invariant-based optimal fitting closure approximation for the numerical prediction of flow-induced fiber orientation," *Journal of Rheology,* vol. 46, pp. 169-194, Jan-Feb 2002.

[101] D. A. Jack and D. E. Smith, "The effect of fibre orientation closure approximations on mechanical property predictions," *Composites Part a-Applied Science and Manufacturing,* vol. 38, pp. 975-982, 2007.

[102] D. A. Jack and D. E. Smith, "Sixth-order fitted closures for short-fiber reinforced polymer composites," *Journal of Thermoplastic Composite Materials,* vol. 19, pp. 217-246, Mar 2006.

[103] D. A. Jack and D. E. Smith, "An invariant based fitted closure of the sixth-order orientation tensor for modeling short-fiber suspensions," *Journal of Rheology,* vol. 49, pp. 1091-1115, Sep-Oct 2005.

[104] D. H. Chung and T. H. Kwon, "Numerical studies of fiber suspensions in an axisymmetric radial diverging flow: the effects of modeling and numerical assumptions," *Journal of Non-Newtonian Fluid Mechanics,* vol. 107, pp. 67-96, Dec 6 2002.

[105] M. P. Petrich, D. L. Koch, and C. Cohen, "An experimental determination of the stress-microstructure relationship in semi-concentrated fiber suspensions," *Journal of Non-Newtonian Fluid Mechanics,* vol. 95, pp. 101-133, Dec 25 2000.

[106] S. G. Advani and E. M. Sozer, *Process modeling in composites manufacturing*. New York: Marcel Dekker, 2003.

[107] R. S. Bay and C. L. Tucker, "Fiber Orientation in Simple Injection Moldings .1. Theory and Numerical-Methods," *Polymer Composites,* vol. 13, pp. 317-331, Aug 1992.

[108] K. Chiba, K. Yasuda, and K. Nakamura, "Numerical solution of fiber suspension flow through a parallel plate channel by coupling flow field with fiber orientation distribution," *Journal of Non-Newtonian Fluid Mechanics,* vol. 99, pp. 145-157, Jul 2001.

[109] M. Yamanoi, J. Maia, and T. S. Kwak, "Analysis of rheological properties of fibre suspensions in a Newtonian fluid by direct fibre simulation. Part 2: Flexible fibre suspensions," *Journal of Non-Newtonian Fluid Mechanics,* vol. 165, pp. 1064-1071, Oct 2010.

[110] M. Yamanoi and J. M. Maia, "Analysis of rheological properties of fibre suspensions in a Newtonian fluid by direct fibre simulation. Part1: Rigid fibre suspensions," *Journal of Non-Newtonian Fluid Mechanics,* vol. 165, pp. 1055-1063, Oct 2010.

[111] D. G. Baird and D. I. Collias, *Polymer processing: principles and design*: Wiley, 1998.

[112] G. K. Batchelor, *An introduction to fluid dynamics*. Cambridge,: U.P., 1967.

[113] M. Altan, S. Subbiah, S. I. Guceri, and R. B. Pipes, "Numerical Prediction of Three-Dimensional Fiber Orientation in Hele-Shaw Flows," *Polymer Engineering and Science,* vol. 30, pp. 848-859, 1990.

[114] K. H. Han and Y. T. Im, "Numerical simulation of three-dimensional fiber orientation in injection molding including fountain flow effect," *Polymer Composites,* vol. 23, pp. 222-238, Apr 2002.

[115] K. K. Kabanemi, J. F. Hetu, and A. GarciaRejon, "Numerical simulation of the flow and fiber orientation in reinforced thermoplastic injection molded products," *International Polymer Processing,* vol. 12, pp. 182-191, Jul 1997.

[116] J. A. Dantzig and C. L. Tucker, *Modeling in Materials Processing*: Cambridge University Press, 2001.

[117] C. A. Hieber and S. F. Shen, "A Finite-Element-Finite-Difference Simulation of the Injection-Molding Filling Process," *Journal of Non-Newtonian Fluid Mechanics,* vol. 7, pp. 1-32, 1980.

[118] E. K. Henry, S.;Kennedy P., "Fiber orientation and the mechanical properties of SFRP parts.," presented at the SPE Tech. Papers, 1994.

[119] C. S. C. Randall, H.H., "Applications of fiber orientation analysis in injection molding of fiber-filled composites," presented at the SPE Tech. Papers, 1994.

[120] H. H. Chiang, C. A. Hieber, and K. K. Wang, "A Unified Simulation of the Filling and Postfilling Stages in Injection-Molding .1. Formulation," *Polymer Engineering and Science,* vol. 31, pp. 116-124, Jan 1991.

[121] P. Filz, "Simulation of the Injection-Molding Process for Crosslinking Materials - Current Situation and Future Possibilities," *Kunststoffe-German Plastics,* vol. 79, pp. 1057-1061, Oct 1989.

[122] A. D. Couinot, L.; Hansen, O.; Dupret, F., "A finite element method for simulating injection molding of thermoplastics," in *Numiform*. vol. 89, J. F. Thompson, Ed., ed Rotterdam, 1989.

[123] S. Subbiah, D. L. Trafford, and S. I. Guceri, "Non-Isothermal Flow of Polymers into Two-Dimensional, Thin Cavity Molds - a Numerical Grid Generation Approach," *International Journal of Heat and Mass Transfer,* vol. 32, pp. 415-434, Mar 1989.

[124] J. Ko and J. R. Youn, "Prediction of Fiber Orientation in the Thickness Plane during Flow Molding of Short-Fiber Composites," *Polymer Composites,* vol. 16, pp. 114-124, Apr 1995.

[125] W. Rose, "Fluid-Fluid Interfaces in Steady Motion," *Nature,* vol. 191, p. 242, 1961.

[126] Z. Tadmor, "Molecular Orientation in Injection Molding," *Journal of Applied Polymer Science,* vol. 18, 1974.

[127] C. G. Gogos, C. F. Huang, and L. R. Schmidt, "The Process of Cavity Filling Including the Fountain Flow in Injection-Molding," *Polymer Engineering and Science,* vol. 26, pp. 1457-1466, Nov 1986.

[128] D. J. Coyle, J. W. Blake, and C. W. Macosko, "The Kinematics of Fountain Flow in Mold-Filling," *Aiche Journal,* vol. 33, pp. 1168-1177, Jul 1987.

[129] H. Mavridis, A. N. Hrymak, and J. Vlachopoulos, "The Effect of Fountain Flow on Molecular-Orientation in Injection-Molding," *Journal of Rheology,* vol. 32, pp. 639-663, Aug 1988.

[130] R. C. Givler, M. J. Crochet, and R. B. Pipes, "Numerical Prediction of Fiber Orientation in Dilute Suspensions," *Journal of Composite Materials,* vol. 17, pp. 330-343, 1983.

[131] A. N. Alexandrou and A. Ahmed, "Unsteady-Flow of Dilute Fiber Suspensions Using a Generalized Eulerian-Lagrangian Approach," *Polymer Composites,* vol. 15, pp. 418-426, Dec 1994.

[132] J. Wang, C. A. Silva, J. C. Viana, F. W. J. van Hattum, A. M. Cunha, and C. L. Tucker, "Prediction of fiber orientation in a rotating compressing and expanding mold," *Polymer Engineering and Science,* vol. 48, pp. 1405-1413, Jul 2008.

[133] H. H. Chiang, C. A. Hieber, and K. K. Wang, "A Unified Simulation of the Filling and Postfilling Stages in Injection-Molding .2. Experimental-Verification," *Polymer Engineering and Science,* vol. 31, pp. 125-139, Jan 1991.

[134] F. Dupret and L. Vanderschuren, "Calculation of the temperature field in injection molding," *Aiche Journal,* vol. 34, pp. 1959-1972, 1988.

[135] S. Ranganathan and S. G. Advani, "A simultaneous solution for flow and fiber orientation in axisymmetric diverging radial flow," *Journal of Non-Newtonian Fluid Mechanics,* vol. 47, pp. 107-136, 1993.

[136] M. Vincent, E. Devilers, and J. F. Agassant, "Fibre orientation calculation in injection moulding of reinforced thermoplastics," *Journal of Non-Newtonian Fluid Mechanics,* vol. 73, pp. 317-326, Dec 1997.

[137] B. E. VerWeyst and C. L. Tucker, "Fiber suspensions in complex geometries: Flow/orientation coupling," *Canadian Journal of Chemical Engineering,* vol. 80, pp. 1093-1106, Dec 2002.

[138] J. M. Park and T. H. Kwon, "Nonisothermal Transient Filling Simulation of Fiber Suspended Viscoelastic Liquid in a Center-Gated Disk," *Polymer Composites,* vol. 32, pp. 427-437, Mar 2011.

[139] B. N. Nguyen, S. K. Bapanapalli, J. D. Holbery, M. T. Smith, V. Kunc, B. J. Frame*, et al.*, "Fiber length and orientation in long-fiber injection-molded thermoplastics - Part I: Modeling of microstructure and elastic properties," *Journal of Composite Materials,* vol. 42, pp. 1003-1029, May 2008.

[140] K. C. Ortman, "Assessing an Orientation Model and Stress Tensor for Semi-Flexible Glass Fibers in Polypropylene Using a Sliding Plate Rheometer: for the Use of Simulating Processes," Ph.D., Chemical Engineering, Virginia Tech, Blacksburg, VA., 2011.

[141] M. C. Altan, S. I. Guceri, and R. B. Pipes, "Anisotropic Channel Flow of Fiber Suspensions," *Journal of Non-Newtonian Fluid Mechanics,* vol. 42, pp. 65-83, Mar 1992.

[142] S. T. Chung and T. H. Kwon, "Numerical-Simulation of Fiber Orientation in Injection-Molding of Short-Fiber-Reinforced Thermoplastics," *Polymer Engineering and Science,* vol. 35, pp. 604-618, Apr 1995.

[143] S. Henmi and N. Mori, "Effects of abrupt expansion geometries on flow-induced fiber orientation and concentration distributions in slit channel flows of fiber suspensions," *Polymer Composites,* vol. 26, pp. 660-670, 2005.

[144] A. Baloch and M. F. Webster, "A Computer-Simulation of Complex Flows of Fiber Suspensions," *Computers & Fluids,* vol. 24, pp. 135-151, Feb 1995.

[145] S. Zhang and J. A. Olson, "Computing orientation distribution and rheology of turbulent fiber suspensions flowing through a contraction," *Engineering Computations,* vol. 24, pp. 52-76, 2007.

[146] K. Chiba and K. Nakamura, "Numerical solution of fiber suspension flow through a complex channel," *Journal of Non-Newtonian Fluid Mechanics,* vol. 78, pp. 167-185, Aug 1998.

[147] J.-z. Lin, "Fiber Orientation Distributions in Slit Channel Flows with Abrupt Expansion for Fiber Suspensions," *Journal of hydrodynamics. Series B,* vol. 20, pp. 696-705, 2008.

[148] R. A. Keiller and E. J. Hinch, "Corner flow of a suspension of rigid rods," *Journal of Non-Newtonian Fluid Mechanics,* vol. 40, pp. 323-335, 1991.

[149] J. Lin, S. Zhang, and W. Zhang, "Numerical research on the fiber suspensions in a turbulent T-shaped branching channel flow," *Chinese Journal of Chemical Engineering,* vol. 15, pp. 30-38, 2007.

[150] K. Ortman, D. Baird, P. Wapperom, and A. Aning, "Prediction of fiber orientation in the injection molding of long fiber suspensions," *Polymer Composites,* vol. 33, pp. 1360-1367, 2012.

Chapter 3

Initial Conditions for Simulating Glass Fiber Orientation in the Filling of Center-Gated Disks

Preface

This chapter describes the simulation of long, semi-flexible fibers in a center-gated disk using both a conventional rigid fiber orientation model and a semi-flexible fiber model which takes into account fiber flexing when calculating orientation. This chapter is organized as a manuscript for publication in Composites Part A: Applied Science and Manufacturing.

3 Initial Conditions for Simulation of Glass Fiber Orientation in the Filling of Center-Gated Disks

Kevin J. Meyer[1], John T. Hofmann[2], and Donald G. Baird[1]*

1. Department of Chemical Engineering, Virginia Tech, Blacksburg, VA 24061
2. Macromolecule and Interfaces Institute, Virginia Tech, Blacksburg, VA 24061

* Corresponding Author: Tel. +1 540 231 5998 ; Fax +1 540 231 2732.
Email Address: dbaird@vt.edu (D.G. Baird)
Present Address:
Department of Chemical Engineering
133 Randolph Hall
Virginia Tech

Blacksburg, VA 24061

3.1 Abstract

This work is concerned with the effect that initial conditions play in simulating long (> 1 mm) glass fiber (LGF) orientation in the filling of a center-gated disk (CGD). For the CGD, most orientation simulations begin at the gate and make assumptions about the initial fiber orientation entering the mold. This paper reports on a method for simulating LGF orientation in a CGD by simulating the sprue, gate, and mold (S-G-M) as a single domain. The velocity field solution is determined using a finite element method including the advancing front. To predict LGF orientation, rigid and flexible fiber models are employed using parameters obtained from rheology. It is observed that predicting LGF orientation in systems with either model is highly sensitive to the choice of initial conditions. Furthermore, the flexible fiber model is observed to be more successful at predicting LGF orientation based on agreement with experimental results.

3.2 Introduction

The flexibility of a glass fiber is used to describe the fiber's tendency to bend in the presence of flow. Switzer and Klingenberg [1] quantified the effective flexibility of a fiber in a viscous medium by proposing the dimensionless group F^{eff}, defined below:

$$F^{eff} = \frac{64\eta_m \dot{\gamma} a_r^4}{E_Y \pi} \tag{3.1}$$

In Eq. (3.1), η_m is the matrix viscosity, $\dot{\gamma}$ is the shear rate, a_r is the aspect ratio ($a_r = L/d$ where L is the length and d is the diameter of the fiber), and E_Y is the Young's modulus. The fiber flexibility is observed to increase with increasing aspect ratio for a given material (fixed Young's modulus, E_Y, and matrix viscosity, η_m). In the literature [2] long glass fibers are defined as any

fiber over the length of 1 mm $\left(F^{eff} > 51\right)$ while short glass fibers are considered under 1 mm and rigid $\left(F^{eff} < 51\right)$. The authors recognize that a number of factors goes into the calculation of the effective fiber flexibility parameter and have chosen to hold all but the fiber length constant for the study presented here.

Modeling glass fiber orientation during injection molding has typically been done by some variant of the Folgar-Tucker (rigid fiber) model which is a modified form of Jeffery's equation for prolate spheroids in a viscous medium [3, 4]. Folgar and Tucker added an additional term, called the isotropic rotary diffusion term, which accounts for the fiber-fiber interaction in a concentrated fiber system. The FT model has shown good agreement in some cases with short glass fiber orientation values obtained from injection molding experiments and is thus a popular choice for commercial software packages [5, 6].

Stress growth experiments in the startup of simple shear suggest that the orientation kinetics in concentrated fiber systems ($\phi > a_r^{-1}$, where ϕ is the fiber volume faction and a_r is the aspect ratio) may more slowly evolve than the FT model predicts [7-9]. A slip parameter was suggested to delay the predicted fiber orientation to better agree with experimental orientation data, but this approach results in a non-objective model [10]. The slip parameter is an addition to further retard the orientation predictions resulting from additional fiber-fiber interaction not accounted for by the rotary diffusion term. An alternative to the non-objective model is the reduced strain closure (RSC) proposed by Wang et al. [11], whereby the slip parameter was moved into the closure approximation generating an eigenvalue problem and maintaining model objectivity. Even though the FT model with the slip parameter is non-objective, it has shown to be useful in describing fiber orientation kinetics in simple flow fields as well as center-gated disks [3, 5, 8, 12, 13].

In glass fiber suspensions, where fibers can be considered flexible, further modification of the theory for fiber orientation may be necessary because of the flexing and deflection that can be observed in these systems. The Bead-Rod (semi-flexible) model for dilute suspensions was introduced by Strautins and Latz [14] as a method to quantify fiber bending during processing. The model was extended to concentrated systems by Ortman et al. [15] with the addition of the slip parameter from Eberle et al. [10] and the isotropic rotary diffusion term from the FT model. The BR model has been shown to agree well with experimental orientation data for simple shear fields [15].

The simulations to calculate fiber orientation utilizing either the rigid fiber model or semi-flexible fiber model are highly dependent upon the choice of values for the model parameters. The parameters for simulation have typically been obtained by empirical relationships or from injection molding experiments [11, 16]. A rheological method to determine these parameters has been presented for both SGF and LGF systems in simple, well controlled flow fields [9, 15]. Eberle et al. [9] showed that the startup of simple shear flow using a "donut" shaped sample in a rotational rheometer can be used to obtain the empirical parameters for calculating fiber orientation in SGF systems. Similarly, Ortman et al. [15, 17] showed that the empirical parameters can be estimated for LGF systems by startup of simple shear using a sliding plate rheometer for both the FT and BR fiber orientation models.

Simulations of short glass fiber (SGF) orientation in center-gated disks have typically begun near the gate, after the fluid turns to enter the mold. At this position, initial conditions are imposed as either assumed values (i.e. completely random, random in the plane, parallel to the flow field, etc.) or experimentally measured values that are obtained from the gate (0% line) after the mold has been filled. The assumption utilized in this technique is commonly referred to the Hele-Shaw approximation and is utilized by many commercial software packages to compute fiber orientation. Bay and Tucker [18] assumed a random orientation at the gate and showed reasonable results using the Hele-Shaw approximation. Vélez-García et al. [10] reported that, on average, the fiber orientation entering the mold is planar random but the orientation did vary across the thickness. Vélez-García et al. [12] also observed that the choice of initial conditions for fiber orientation at the mold entrance persisted to approximately 50% of the mold. Chung and Kwon [19] noted the contribution of the sprue region of the center-gated disk to the predictions of fiber orientation in the mold. It was reported that all of the components of orientation for short glass fibers were influenced by where the solution of orientation was started (inlet of gate vs. inlet of sprue).

Little work has been performed in the prediction of orientation values for long glass fibers in center-gated disks. Ortman et al. [20] used experimentally determined values at the inlet to the mold. Orientation predictions were carried out with a rigid fiber model and a semi-flexible fiber model and both models showed predictions in reasonable agreement with experimental data up to 50% of the mold fill.

The goal of this work is to evaluate how the choice of initial conditions affects the prediction of long glass fiber orientation in a center-gated disk. Common assumptions about fiber orientation at the gate are compared to a method of simulating the sprue, gate, and mold (S-G-M) as a single domain [19]. The S-G-M method starts from the beginning of the sprue with an initially random fiber orientation and allows the flow field, including the advancing front, to completely dictate fiber orientation. The solution for velocity fields and fiber orientation are decoupled so that a two-step process to calculate orientation is invoked. The velocity field is solved using a non-isothermal volume of fluid method (VOF) in the ANSYS® Polyflow software package. Fiber orientation is calculated using MATLAB routines developed in our laboratory utilizing the finite difference method. The quality of each initial condition choice is evaluated against experimental orientation data obtained from a center-gated disk. Furthermore, a rigid and a flexible fiber model are used in predicting LGF orientation with conclusions drawn as to the accuracy of each model.

3.3 Governing Equations

In the following section, the pertinent equations for the solution of fiber orientation are presented. First the equations of motion and energy are presented for the solution of the neat polymer matrix. Next, the rigid fiber model is described for concentrated fiber systems utilizing a slip parameter. Finally, the semi-flexible fiber model is presented as an alternative to model concentrated a fiber system that attempts to account for flexibility sometimes seen in long glass fiber systems.

3.3.1 Equations of Motion

The fluid is assumed to be incompressible so that the mass and momentum equations take on the familiar form of Eq. (3.2) and Eq. (3.3), respectively.

$$0 = \nabla \cdot \bar{v} \tag{3.2}$$

$$0 = -\nabla P + \nabla \cdot \bar{\bar{\tau}} \tag{3.3}$$

In Eq. (3.2) and Eq. (3.3), $\bar{v}$ is the velocity, P is the pressure, and $\bar{\bar{\tau}}$ is the extra stress tensor due to the deforming polymeric fluid. The flow was assumed laminar after calculation of the Reynolds number showed that $\mathrm{Re} \approx 10^{-3}$ throughout the cavity. The Carreau model was used to account for the shear thinning behavior of the neat matrix and is given below in Eq. (3.4) where η_0 zero shear viscosity, λ is a parameter which governs the onset of shear thinning, and n is a parameter governing the degree of shear thinning of the matrix.

$$\eta_m(\dot{\gamma}) = \eta_0 \left[1 + (\lambda \dot{\gamma})^2 \right]^{\frac{n-1}{2}} \tag{3.4}$$

Finally, the fluid is assumed to be homogenous within the domain. In fiber suspensions, especially containing long fibers, the polymer fluid filling the cavity has voids, etc. By simulating only the matrix viscosity the assumption of a homogenous flow field is more accurate and thus adopted here.

3.3.2 Thermal Effects

Minimizing thermal gradients in the molding operation was achieved through the use of fast mold fill times. To corroborate the reduction of thermal effects in the experimental injection moldings, the molding simulations were initially performed under non-isothermal conditions to account for the thermal gradients observed in injection molding. The result of the non-isothermal simulations in the mold showed only a $6K$ temperature drop from the center of the disk to the wall $(\approx 1 \text{ mm})$ of the mold and only a $10K$ temperature drop very near the wall $(\approx 0.1 \text{ mm})$. Additionally, the temperature drop observed from the center of the sprue to the wall $(\approx 0.75 \text{ mm})$ was less than $2K$. This small temperature gradient in the mold allowed for the isothermal assumption in all subsequent simulations. Hence, the viscosity was assumed independent of temperature in the melt.

3.3.3 Equations of Rigid Fiber Orientation

Quantifying rigid fiber orientation in a concentrated suspension is accomplished through the use of orientation tensors given by Advani and Tucker [21]. The orientation distribution function, ψ, describes the probability of a single fiber's orientation within (θ, ϕ) and $(\theta + d\theta, \phi + d\phi)$. The 2nd and 4th moments of the orientation tensor can be obtained by evaluating the integrals shown in Eq. (3.5) and Eq. (3.6), respectfully:

$$\bar{\bar{A}} = \iint \overline{pp}\ \psi(\bar{p}, t)\, d\bar{p} \tag{3.5}$$

$$\bar{\bar{A}}_4 = \iint \overline{pppp}\ \psi(\bar{p}, t)\, d\bar{p} \tag{3.6}$$

The 4th order tensor in Eq. (3.6) requires the use of a closure approximation that reduces the 4th order tensor to a combination of 2nd order tensors. Advani and Tucker recognized the closure problem [21] and proposed closure approximations [22] reduce the fourth-order tensor and a number of closure approximations are available which are examined in Chung and Kwon [23].

For this work the invariant-based optimal fitting (IBOF) closure is used to estimate the 4th order orientation tensor given by Chung and Kwon [24] which is based on polynomial expansions of the 2nd and 3rd invariants of the 2nd order orientation tensor, $\overline{\overline{A}}$.

Modeling glass fiber orientation in concentrated suspensions has been successful through the use of the rigid fiber model [3, 5, 18] which is a modification of Jeffery's equation for prolate spheroids in a viscous medium [4]. To more accurately reflect the slower orientation kinetics seen in experimental data of concentrated glass fiber suspensions, Sepehr et al. [8] and Huynh [25] proposed adding a phenomenological slip parameter, α, to the original rigid fiber model. The modified model is shown in Eq. (3.7) where $\overline{\overline{W}}$ is the vorticity tensor $\left(\overline{\overline{W}} = \frac{1}{2}\left[(\nabla\overline{v})^T - \nabla\overline{v}\right]\right)$, $\overline{\overline{D}}$ is the rate of deformation tensor $\left(\overline{\overline{D}} = \frac{1}{2}\left[(\nabla\overline{v})^T + \nabla\overline{v}\right]\right)$, $\dot{\gamma}$ is the magnitude of the shear rate $\left(\dot{\gamma} = \sqrt{\frac{1}{2}\overline{\overline{D}} : \overline{\overline{D}}}\right)$, ξ is the shape factor $\xi = a_r^2 + 1/a_r^2 - 1$, $\overline{\overline{I}}$ is the identity tensor and $\nabla\overline{v} = \partial v_j / \partial x_i$:

$$\frac{D\overline{\overline{A}}}{Dt} = \alpha\left[\overline{\overline{W}}\cdot\overline{\overline{A}} - \overline{\overline{A}}\cdot\overline{\overline{W}} + \xi\left(\overline{\overline{D}}\cdot\overline{\overline{A}} + \overline{\overline{A}}\cdot\overline{\overline{D}} - 2\overline{\overline{A}}_4 : \overline{\overline{D}}\right) + 2C_I\dot{\gamma}\left(\overline{\overline{I}} - 3\overline{\overline{A}}\right)\right] \tag{3.7}$$

The shape factor, ξ, for glass fiber systems is usually given a value of unity since a_r is generally large for glass fibers $(\approx 10-100)$. The third term on the right hand side of Eq. (3.7) is the isotropic rotary diffusion term, which accounts for the fiber-fiber interaction inherent to the concentrated suspension and is loosely based on the term for isotropic diffusivity associated with Brownian rods [26]. The C_I term is an empirical parameter, which accounts for some of the of fiber-fiber interaction in a concentrated suspension. The slip coefficient, α, is another empirical parameter with a value between 0 and 1 which tries to account for the slower fiber orientation kinetics seen in concentrated glass fiber systems. The model given by Eq. (3.7) has been chosen to simulate fiber orientation because it is seen to qualitatively agree with experimental data in both simple shear flow and center-gated disks [9, 12, 15].

3.3.4 Equations of Flexible Fiber Orientation

As fibers increase in length, the orientation model representing those fibers may need to be able to account for the bending and deflection that is sometimes encountered. The semi-flexible model was first proposed by Strautins and Latz [14] as a method to predict flexible fiber orientation in a dilute suspension using the compact tensor notation described above [21]. The flexible fiber was represented as two vectors, $\overline{p}$ and $\overline{q}$, connected by a ball and socket joint with an internal resistivity to bending shown in Figure 3.2. The semi-flexible system uses the similar tensor notation as the rigid fiber model but, because of the two-rod system, three moments occur and are defined by Eqs. (3.8) - (3.10):

$$\overline{\overline{A}} = \iint \overline{p}\overline{p}\ \psi(\overline{p},\overline{q},t)\,d\overline{p}\,d\overline{q} \tag{3.8}$$

$$\overline{\overline{B}} = \iint \overline{p}\overline{q}\ \psi(\overline{p},\overline{q},t)\,d\overline{p}\,d\overline{q} \tag{3.9}$$

$$\overline{C} = \iint \overline{p}\ \psi(\overline{p},\overline{q},t)\,d\overline{p}d\overline{q} \tag{3.10}$$

The three moments of the semi-flexible system give rise to a set of three orientation equations, which have been extended by Ortman et al. [17] to concentrated solutions by adding the isotropic rotary diffusion term from Folgar and Tucker in Eqs. (3.11) - (3.14):

$$\frac{D\overline{\overline{A}}}{Dt} = \alpha\left[\left(\overline{\overline{W}}\cdot\overline{\overline{A}} - \overline{\overline{A}}\cdot\overline{\overline{W}}\right) + \xi\left(\overline{\overline{D}}\cdot\overline{\overline{A}} + \overline{\overline{A}}\cdot\overline{\overline{D}} - 2\overline{\overline{D}}:\overline{\overline{A}}_4\right) + \ldots\right.$$
$$\left.\frac{l_b}{2}\left[\overline{C}\overline{m} + \overline{m}\overline{C} - 2\left(\overline{m}\cdot\overline{C}\right)\overline{\overline{A}}\right] + 2k\left[\overline{\overline{B}} - \overline{\overline{A}}\ \mathrm{tr}\left(\overline{\overline{B}}\right)\right] - 6C_I\dot{\gamma}\left(\overline{\overline{A}} - \tfrac{1}{3}\overline{\overline{I}}\right)\right] \tag{3.11}$$

$$\frac{D\overline{\overline{B}}}{Dt} = \alpha\left[\left(\overline{\overline{W}}\cdot\overline{\overline{B}} - \overline{\overline{B}}\cdot\overline{\overline{W}}\right) + \xi\left(\overline{\overline{D}}\cdot\overline{\overline{B}} + \overline{\overline{B}}\cdot\overline{\overline{D}} - 2\left(\overline{\overline{D}}:\overline{\overline{A}}\right)\overline{\overline{B}}\right) + \ldots\right.$$
$$\left.\frac{l_b}{2}\left[\overline{C}\overline{m} + \overline{m}\overline{C} - 2\left(\overline{m}\cdot\overline{C}\right)\overline{\overline{A}}\right] + 2k\left[\overline{\overline{A}} - \overline{\overline{B}}\ \mathrm{tr}\left(\overline{\overline{B}}\right)\right] - 4C_I\dot{\gamma}\overline{\overline{B}}\right] \tag{3.12}$$

$$\frac{D\overline{C}}{Dt} = \alpha\left[\nabla\overline{v}^T\cdot\overline{C} - \left(\overline{\overline{A}}:\nabla\overline{v}^T\right)\overline{C} + \frac{l_b}{2}\left[\overline{m} - \overline{C}\left(\overline{m}\cdot\overline{C}\right)\right] - k\overline{C}\left[1 - tr\left(\overline{\overline{B}}\right)\right] - 2C_I\dot{\gamma}\overline{C}\right] \tag{3.13}$$

$$\overline{m} = \sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3}\frac{\partial^2 v_i}{\partial x_j\,\partial x_k}A_{jk}\overline{\delta}_i \tag{3.14}$$

The internal resistivity to bending, k, is a coefficient that attempts to account for the physical bending that may occur in the system of fibers. As the value of k increases, the semi-flexible model behaves more like the rigid fiber model and in the limit of $k \to \infty$ the semi-flexible model parallels the rigid fiber model. Conversely, as the value of k decreases the fiber becomes much more flexible. In the equations presented above, the flexibility of the fiber is initiated in the hydrodynamic contributions given by Eq. (3.14) and is due to the bending that may occur do to gradients in the flow field.

From these expressions an end-to-end vector is defined which establishes the "average" orientation of a slightly bent fiber. This vector is also formulated as a tensor by integrating the orientation distribution function with the result being Eq.(3.15):

$$\overline{r} = \iint l_b^2\left(\overline{p} - \overline{q}\right)\left(\overline{p} - \overline{q}\right)\psi\left(\overline{p},\overline{q},t\right)d\overline{p}\,d\overline{q} \tag{3.15}$$

If the tensor in Eq. (3.15) is normalized, then the $\overline{\overline{R}}$ tensor is produced $\left(\overline{\overline{R}} = \langle\overline{rr}\rangle\big/\mathrm{tr}\left(\overline{r}\right)\right)$ and is related to the $\overline{\overline{A}}$ and $\overline{\overline{B}}$ tensors through the relation shown in Eq.(3.16):

$$\overline{\overline{R}} = \frac{\overline{\overline{A}} - \overline{\overline{B}}}{1 - \mathrm{tr}\left(\overline{\overline{B}}\right)} \tag{3.16}$$

In the limit when no fiber flexing occurs, the $\overline{\overline{B}}$ tensor becomes small and the $\overline{\overline{A}}$ tensor is recovered and behaves like rigid fiber theory.

3.3.5 Use of Three-Dimensional Orientation Tensors

The center-gated disk geometry chosen for these studies was based on the industrial application of long glass fibers in a radially diverging flow field. Taking into account the average fiber length of the system and the gaps with which these fibers must flow through, it is easy to see that certain fiber orientations are precluded from ever occurring. Three dimensional orientation tensors are used in this work because the technique of simulating the sprue, gate and mold region requires it. For example, a long glass fiber, upon entering the sprue, will most likely have a large value for the global "2" orientation component and be limited from ever fully being oriented in the global "1" or "3" directions because the radius of the sprue is two small. Once this same fiber turned to enter the mold, the global "2" component would never be fully reach again because of the thickness of the mold and the length of the fiber. If either of these two cases were calculated separately than a two-dimensional orientation tensor or possibly some other representation could be used. In this work, the entire domain is simulated in a continuous manner thus requiring a global three-dimensional orientation tensor.

3.4 Experimental

3.4.1 Injection Molding Conditions

Injection molded center-gated discs were created for experimental evaluation of the orientation tensor using 30 wt% long glass fiber (13 mm) reinforced polypropylene provided by SABIC Innovative Plastics. The temperatures of the feed, compression, and metering zones within the injection molding machine (Arburg Allrounder, Model 221-55-250) were set to 190, 210, and 210 °C, respectively, while the mold temperature was held constant at 79 °C. The center-gated discs have a radius of 89.3 mm and thickness of 1.98 mm. The sprue was 65 mm in length with an initial radius of 1.45 mm and a radius at the gate of 1.75 mm. The disks were molded with a fill time of 2.00 seconds and a backpressure of approximately 20 MPa. To ensure that the machine was operating in an equilibrium state, the first ten discs molded were discarded before selecting samples for analysis. Additionally, each disc was left in the mold for a period of 20 minutes prior to removal in an effort to reduce warping.

Stress relaxation tests were performed on the suspension in simple shear to observe the effect of fibers relaxing during the period of time that the sample is left in the mold to reduce warping. It was found that any relaxation of the sample occurs within seconds of the molding operation. Furthermore, upon investigation of experimental data, there are still are large population of fibers bent suggesting that the fibers do not relax during the holding time in the mold.

3.4.2 Determination of Fiber Length Distribution

The Fiber Length Distribution (FLD) is determined in order to ascertain the average fiber length and to ensure that the injection molded discs contained a truly long glass fiber population. Methods described by Nguyen et al. [27] were used to measure the FLD of a population of approximately 2,000 fibers. In brief, this method involves selecting a large portion of an injection-molded part and burning off the polymer matrix in a high temperature furnace, leaving a mat of glass fibers behind. A small amount of epoxy resin is applied to the fiber mat and allowed to cure, and all excess fibers not encapsulated by the epoxy are carefully removed and discarded. The remaining sampling region is returned to the furnace in order to burn off the epoxy resin. Finally, the remaining glass fibers are separated and measured by image analysis to construct the FLD. By using this method, the post-processing number average fiber length and

diameter is determined to be 3.90 mm (± 0.01mm) and 14.5 μm, respectively. The resulting FLD is asymmetric, exhibits a long tail in the long fiber length regime, and appears to be represented by a Weibull function, as previously reported in the literature [27, 28]. As such, it can be concluded that the majority of fibers in the disc possess lengths greater than 1 mm and that the population consists primarily of long semi-flexible fibers.

3.4.3 Orientation Measurements

Representative short shot discs stopped at 90% of the mold fill were chosen for analysis at 0, 10, 40, and 90% of mold fill. The selected samples were prepared by polishing and plasma etching, using the method described in detail by Vélez-García et al. [29]. After preparation, each inspection line was imaged at 20X magnification using an optical microscope with a motorized stage and image stitching software (Nikon Eclipse LV100, NIS-Elements Basic Research Software, version 3.10). The resulting images had a width of 700 μm and height equal to the thickness of the disc, and included a minimum of 350 fibers per inspection line. The unambiguous components of the orientation tensor were subsequently determined using the traditional Method of Ellipses (MoE) [30, 31].

Prior to discussing the actual orientation data, however, it is worth discussing the validity of the traditional MoE for application to long fiber systems. While the traditional MoE is based upon a rigid-rod assumption, the increased flexibility possible with long fibers presents inherent difficulties in evaluation of orientation. However, the large theta-theta component of the rate of deformation tensor, $\dot{\gamma}_{\theta\theta}$, seen in the selected geometry combined with the axisymmetric nature of the center-gated disc results in a population of fibers aligned predominantly in the transverse-to-flow direction (i.e. in the theta direction). These results, detailed by Hofmann et al. [31], suggest that no modifications to the method are necessary for the selected geometry and fiber length. It is worth noting, however, that such a fortuitous result is not expected to translate to more complicated parts where the axisymmetric benefit is lost and a 3-dimensional flow dominates.

3.4.4 Determination of Flexibility Parameter

The flexibility parameter for each fiber, k_i , was estimated through Eq. (3.17) based on basic mechanical analysis of a simply supported beam.

$$k_i = \left(\frac{E_Y}{64\eta_m} \right) \frac{d^3}{l_b^3} \tag{3.17}$$

The value for each fiber was then average to determine an average flexibility parameter based on Eq. (3.18).

$$k = \frac{\sum_i n_i k_i}{\sum_i n_i} \tag{3.18}$$

3.5 Problem Formulation

3.5.1 Solution Process

The domain was divided into two separate meshes and merged before solving for the velocity fields. The sprue was meshed using 10×31 and the mold was meshed using 31×200

both utilizing linear rectangular elements. Steep gradients at the wall are captured by using unequal grid spacing and no further increase in numerical accuracy is seen by increasing the domain discretization either through the thickness or length of the mold.

Tucker [32] first introduced a simplification for fiber suspensions flowing in narrow gaps where the extra stress depends on the velocity gradients but not the fiber orientation. This fiber simulation technique is known as the decoupled approach and has been verified by a number of authors [12, 16, 18, 33-35] and is the solution approach adopted for the simulations in this work. Further justification for using the decoupled approach is the lack of a valid stress tensor for concentration fiber suspensions. Forms of the stress tensor has been developed and confirmed for dilute [36] While a form of the stress tensor has been developed for dilute and semi-concentrated suspensions and semi-dilute [37] suspensions but no such form exists for concentrated fiber systems that is successful in all flow fields [19]. The velocity fields during the injection molding process are calculated using the ANSYS Polyflow FEM non-Newtonian solver with a time dependent volume of fluid (VOF) method. The parameters for simulating the polymer matrix filling the center-gated mold are shown in Table 3.1.

Upon completion of the FEM simulations, the spacial coordinates and velocity data are imported into MATLAB® (2011b, ver. 7.13.0.564, The Mathworks, Inc., Natick, MA) routines where fiber orientation is calculated. Because the equations of motion and the equations of orientation have been decoupled, the set of hyperbolic PDE's is reduced to a set of highly non-linear ODE's which can be solved using the 'ode15s' function in the MATLAB software package. The routines employ a combination of backward and central finite difference methods to approximate the convective term in the material derivative and the mixed partial derivative in Eq. (3.14).

The two phenomenological constants for simulation of fiber orientation have been determined using a sliding plate rheometer and the technique described in Ortman et al. [17]. The parameters are calculated for 30 wt. % LGF in a polypropylene matrix and are $\alpha = 0.25$ and $C_I = 0.005$ for the rigid fiber model and $\alpha = 0.13$ and $C_I = 0.025$ for the semi-flexible model. Additionally, l_b is one half of the average measured fiber length and $k = 218\ \mathrm{s}^{-1}$ for the semi-flexible fiber suspension of 30 wt. % LGF in a polypropylene matrix.

3.5.2 Boundary Conditions

For the solution of the velocity fields it is necessary to impose some boundary conditions on the simulated domain. The no-slip condition is applied at the walls $(\mathbf{v} = 0, x \pm -h)$. For the air/polymer interface in the mold, the VOF method in ANSYS Polyflow only requires the specification of the normal and tangential forces which are defined to be zero $(\mathbf{f}_n = \mathbf{f}_t = 0)$. The inlet volumetric flow was prescribed as $500\,\mathrm{mm}^3/\mathrm{s}$ and the flow is assumed to be fully developed. The neat matrix simulation was suspended when the cavity was 99% full. It was also assumed that the mold thickness does not play a role in dictating fiber orientation based on the work of Nguyen et al. [38].

3.5.3 Initialization of Orientation Solution

The solution of fiber orientation presented in this work is initialized three different ways. The first type of initialization makes an assumption about fiber orientation at the entrance to the

mold cavity. Two initial configurations that have been applied to the mold entrance with success for glass fiber systems are random [19] and random in the plane [10, 18] shown below by the tensors in Eq. (3.19), respectively:

$$\overline{\overline{A}}_0 = \begin{pmatrix} \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{3} \end{pmatrix}; \quad \overline{\overline{A}}_0 = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix} \tag{3.19}$$

The second initialization approach uses experimental orientation data obtained at the gate. Chung and Kwon [19] , Vélez-García et al. [12], used SGF experimental orientation data obtained from the gate and Ortman et al. [39] used LGF experimental orientation values after the mold had been filled as initial conditions to the mold entrance and observed reasonable agreement with experimental data. The last initialization approach, first proposed by Chung and Kwon [19] for SGF systems, will be started at the entrance to the sprue and allow the fiber orientation to evolve down the sprue, turn through the gate, and fill the center-gated disk mold. The sprue simulation is initialized with a random orientation at the sprue/die interface, as described in Eq. (3.19).

3.5.4 Testing of Numerical Solution

The numerical method to calculate the velocity gradients was tested by comparing the numerically predicted velocities calculated with Polyflow to the analytical solution for a Newtonian fluid in radially diverging flow. Figure 3(a) shows the results of this simulation plotting the radial velocity of the fluid at 40% of the mold fill. The maximum discrepancy between the analytical solution and the numerical solution is less than 0.25%.

The numerical prediction of fiber orientation was also verified by using simple shear flow as a test case. The transient viscosity of the glass fiber suspensions was calculated by using a sliding plate rheometer [17]. The transient viscosity can be solved for easily in simple shear and is plotted along with the experimentally obtained data in Figure 3. The transient viscosity was also calculated using the Polyflow and MATLAB numerical routines with the resulting predictions also plotted in Figure 3(b). It can be seen that the numerical method is very close to that of the analytical solution and well inside the error inherent to these long glass fiber systems [40].

3.6 Results

In the following section, results from the simulations will be discussed. First LGF orientation results from the sprue using the S-G-M method are presented to show that the choice of initial conditions at the sprue entry have been washed out by 80% of the fill of the sprue. Then, results from different assumptions about solution initialization are compared. Finally, results from simulating LGF orientation using the S-G-M method for both a rigid and semi-flexible fiber model are shown.

3.6.1 Simulation of Fiber Orientation in the Sprue of a Center-Gated Disk

The long glass fiber-filled fluid was allowed to fill the sprue to the point where the fluid turned to enter the mold and then the injection was stopped. The qualitative representation of this investigation is shown in Figure 3.5 where a number of regions of interest have been highlighted. Two regions in the sprue were targeted in order to assess the accuracy of the models before the

suspension entered the mold cavity. At position CE60, the CE data is compared against predicted values of the A_{11} and A_{22} (or R_{11} and R_{22} for the semi-flexible model), as is shown in Figure 3.6 (a). Qualitatively both fiber orientation models capture the trends in the sprue with high A_{22} (R_{22}) values at the wall (high shear) and low A_{22} (R_{22}) values in the center. Quantitatively the flexible fiber model predicts fiber orientation more accurately than the rigid fiber model. At postion CE80 shown in Figure 3.6(b), both models perform equally well correctly capturing the A_{11} and A_{22} (or R_{11} and R_{22}) fiber orientation kinetics. Both models qualitatively predict the CE orientation data at 80% of the sprue. It is important to point out that a number of different fiber orientations were assumed at the sprue inlet but all of them provided solutions at positions CE60 and CE80 that were within 5% of one another. From this it can be said that the choice of these initial conditions do not play a role in the mold fiber orientation predictions as the assumptions at the sprue a washed out before they reach the mold cavity.

Often assumptions are made about the orientation of long glass fibers entering the mold cavity to simplify the solution, but these assumed values are incorrect based on what has been seen experimentally. The two sets of experimental data in Figure 3.6 show that the CE A_{11} orientation data and the FE A_{11} orientation data are markedly different at the gate, by as much as 40% at the walls. If the predicted fiber orientation values are compared to the CE and FE data, both the rigid and flexible model follow the wide orientation distribution exhibited by the experimental data well. The CE experimental orientation data is clearly assymetric and that trend is also captured by both orientation models. Using the S-G-M method the rigid fiber model predicts orientation slightly better at the lower wall and the flexible fiber model predicts slightly better at the upper wall. Again using the S-G-M method both of the models are better at predicting the A_{33} (R_{33} for flexible model) component than the A_{11} (R_{11} for flexible model) component and qualitatively capture the assymetry observed at the gate region. Furthermore, using the S-G-M method and comparing the results to experimental orientation data confirm that the choice of either random or planar random initial conditions are observed to be unsatisfactory for simulating LGF filling operations since the trends in Figure 3.6 show decisively that the initial conditions are a function of the mold cavity thickness.

3.6.2 Simulation of Long Glass Fiber Orientation in the Mold of a Center-Gated Disk

The prediction of LGF orientation was also carried out using the two models of interest with four sets of initial conditions. The results of the LGF simulations at the FE10 sampling point for both the FT model and BR model are shown in Figure 3.8 (a) and (b), respectfully. The choice of random initial conditions using the FT model shows a traditional shell and core structure in the cross section, which does not qualitatively agree with the experimental LGF data. The other three initial conditions are observed to more qualitatively agree the experimental data, with the exception of the orientation predictions at the wall. Conversely the BR model, shown in Figure 3.8 (b), does a more encouraging job at predicting LGF orientation in the CGD. The experimental initial conditions as well as the S-G-M method prove to be more successful at capturing the asymmetric nature of the system and quantitatively agree well with the experimental FE data for LGF through the mold including near the walls.

At the FE40 sampling point, the FT model does a poor qualitative job of capturing the kinetics of the experimental FE data, shown in (a). While the choice of initial condition for fiber orientation does affect the predicted values at FE40, none of the choice shown qualitative

agreement with experimentally obtained LGF data. The BR model results, presented in Figure 3.9 (b), show much more encouraging trends overall, with the choice of initial conditions for fiber orientation playing a role in the predicted values at FE40. The random and planar random choices both qualitatively capture the orientation seen in the center of the mold, but fail to predict both the wall orientation and the asymmetric behavior observed in the experimental results. The experimental and S-G-M methods are observed to be more accurate than the assumed initial conditions, in predicting the asymmetric behavior of the LGF both in the center of the cavity and near the walls.

The results from the LGF orientation simulation for the FE80 position are shown in Figure 3.10 (a) and (b) for the FT model and BR model, respectively. Again, the FT model does a poor job using any of the four sets of initial conditions. The FT model is observed to qualitatively predict orientation of LGF near the center of the mold but does a very poor job near the walls. The BR model again does a more encouraging job in predicting LGF orientation. The two assumed initial conditions, random and planar random, are seen to agree qualitatively with the experimental data, but again do not capture the asymmetric nature of the experimentally obtained values at the FE80 position. The experimental initial conditions are the least encouraging, predicting larger fluctuations through the thickness than what is observed experimentally. The S-G-M method performs well predicting the LGF orientation through the thickness with the exception of the walls.

3.6.3 Comparison of Long Glass Fiber Orientation Predictions Using the S-G-M Method

To evaluate the success of either the FT model or BR model at predicting LGF orientation in a CGD, direct comparisons were made using the S-G-M method shown in Figure 3.11 (a), (b), and (c) for the FE10, FE40, and FE80 positions, respectfully. At the FE10 position both of the models qualitatively predict the LGF orientation through the thickness except near the walls. The FT model is seen to quantitatively over-predict fiber orientation through the entire cavity and under predict fiber orientation by as much as 500% for the A_{33} component near the walls of the CGD. The BR model shows predictions for LGF orientation that much more closely agree with experimental data near the wall. At the FE40 position the FT model performs poorly, over predicting LGF orientation near the walls and in the center of the mold using the S-G-M method. The BR model performs markedly better capturing both the orientation of LGF near the walls as well as through the cavity with only slight discrepancies near the top of the mold. At the FE80 position, the FT model qualitatively captures only the center third of the mold cavity, significantly over predicting LGF orientation near the walls using the S-G-M method. The BR model orientation predictions more closely resemble the experimental LGF orientation data but under predict orientation in regions close to the mold walls.

3.7 Conclusions

The choice of initial conditions and how to initialize the solution process to calculate glass fiber orientation was proposed to be an influential part in the predictions generated by current modeling techniques. The study was carried out for long glass fiber systems starting the fiber orientation simulations at both the mold entrance and the sprue entrance and using either random, planar random, or experimental initial orientation conditions at those positions. The choice of initial conditions at the sprue entrance was observed to completely wash out by 80% of the sprue length so that the initial conditions into the mold cavity were governed completely by the flow field. The gate position (0% mold fill) was examined to investigate the accuracy of the

experimental values used to simulate the fiber orientation predictions in the mold. It was shown that the orientation of glass fibers at the gate position as they enter the mold cavity and the orientation of glass fibers after the cavity has been filled are different, most notably at the walls. This provided evidence that the simulations should be carried out from the beginning of the sprue and let the fiber orientation evolve according to the flow field.

Long glass fiber orientation predictions for both the FT model and BR model were observed to be influenced by the choice of fiber orientation initial conditions. For the long fiber orientation simulations the FT model showed poor agreement with experimental data no matter the choice of initialization method. The BR model predictions showed much more encouraging results when compared with experimental data, especially when using the S-G-M method. The FT and BR models were compared at the FE10, FE40, and FE80 positions using the S-G-M method where it was observed that the BR model more accurately predicts experimentally observed long glass fiber orientation.

3.8 Acknowledgments

The financial support for this work is truly appreciated and has been provided by the National Science Foundation, Grant No. CMMI-0853537. We would also like to thank SABIC for providing the materials used in this work. Additional gratitude is given to the Department of Mechanical Engineering and Material Science and Engineering Department at Virginia Tech for the use of facilities. The author would also like to thank Syed Mazahir and Mark Cieslinski for the fruitful discussions.

3.9 References

[1] L. H. Switzer and D. J. Klingenberg, "Rheology of sheared flexible fiber suspensions via fiber-level simulations," *Journal of Rheology,* vol. 47, pp. 759-778, May-Jun 2003.

[2] J. M. Crosby, "Recent Advances in Thermoplastic Composites," *Advanced Materials & Processes,* vol. 133, pp. 56-59, Mar 1988.

[3] F. Folgar and C. L. Tucker, "Orientation Behavior of Rigid Fibers in Concentrated Suspensions," *Journal of Rheology,* vol. 26, pp. 604-604, 1982.

[4] G. B. Jeffery, "The Motion of Ellipsoidal Particles Immersed in a Viscous Fluid," *Proceedings of the Royal Society a-Mathematical Physical and Engineering Sciences,* vol. 100, pp. 161-179, 1922.

[5] R. S. Bay and C. L. Tucker, "Fiber Orientation in Simple Injection Moldings .2. Experimental Results," *Polymer Composites,* vol. 13, pp. 332-341, Aug 1992.

[6] S. T. Chung and T. H. Kwon, "Coupled analysis of injection molding filling and fiber orientation, including in-plane velocity gradient effect," *Polymer Composites,* vol. 17, pp. 859-872, Dec 1996.

[7] M. Sepehr, P. J. Carreau, M. Grmela, G. Ausias, and P. G. Lafleur, "Comparison of rheological properties of fiber suspensions with model predictions," *Journal of Polymer Engineering,* vol. 24, pp. 579-610, Nov-Dec 2004.

[8] M. Sepehr, G. Ausias, and P. J. Carreau, "Rheological properties of short fiber filled polypropylene in transient shear flow," *Journal of Non-Newtonian Fluid Mechanics,* vol. 123, pp. 19-32, Oct 15 2004.

[9] D. G. Baird, A. P. R. Eberle, P. Wapperom, and G. M. Velez-Garcia, "Using transient shear rheology to determine material parameters in fiber suspension theory," *Journal of Rheology,* vol. 53, pp. 685-705, May-Jun 2009.

[10] D. G. Baird, A. P. R. Eberle, G. M. Velez-Garcia, and P. Wapperom, "Fiber orientation kinetics of a concentrated short glass fiber suspension in startup of simple shear flow," *Journal of Non-Newtonian Fluid Mechanics,* vol. 165, pp. 110-119, Feb 2010.

[11] C. L. Tucker, J. Wang, and J. F. O'Gara, "An objective model for slow orientation kinetics in concentrated fiber suspensions: Theory and rheological evidence," *Journal of Rheology,* vol. 52, pp. 1179-1200, Sep-Oct 2008.

[12] G. M. Velez-Garcia, S. M. Mazahir, P. Wapperom, and D. G. Baird, "Simulation of Injection Molding Using a Model with Delayed Fiber Orientation," *International Polymer Processing,* vol. 26, pp. 331-339, Jul 2011.

[13] C. A. Hieber and S. F. Shen, "A Finite-Element-Finite-Difference Simulation of the Injection-Molding Filling Process," *Journal of Non-Newtonian Fluid Mechanics,* vol. 7, pp. 1-32, 1980.

[14] U. Strautins and A. Latz, "Flow-driven orientation dynamics of semiflexible fiber systems," *Rheologica Acta,* vol. 46, pp. 1057-1064, Oct 2007.

[15] K. C. Ortman, P. Wapperom, A. Whittington, and D. G. Baird, "Using startup of steady shear flow in a sliding plate rheometer to determine material parameters for the purpose of predicting long fiber orientation," *Journal of Rheology,* vol. 56, pp. 955-981, 2012.

[16] C. L. Tucker and J. H. Phelps, "An anisotropic rotary diffusion model for fiber orientation in short- and long-fiber thermoplastics," *Journal of Non-Newtonian Fluid Mechanics,* vol. 156, pp. 165-176, Feb 2009.

[17] K. C. Ortman, N. Agarwal, A. P. R. Eberle, D. G. Baird, P. Wapperom, and A. J. Giacomin, "Transient shear flow behavior of concentrated long glass fiber suspensions in a sliding plate rheometer," *Journal of Non-Newtonian Fluid Mechanics,* vol. 166, pp. 884-895, Sep 1 2011.

[18] R. S. Bay and C. L. Tucker, "Fiber Orientation in Simple Injection Moldings .1. Theory and Numerical-Methods," *Polymer Composites,* vol. 13, pp. 317-331, Aug 1992.

[19] D. H. Chung and T. H. Kwon, "Numerical studies of fiber suspensions in an axisymmetric radial diverging flow: the effects of modeling and numerical assumptions," *Journal of Non-Newtonian Fluid Mechanics,* vol. 107, pp. 67-96, Dec 6 2002.

[20] K. Ortman, D. Baird, P. Wapperom, and A. Aning, "Prediction of fiber orientation in the injection molding of long fiber suspensions," *Polymer Composites,* vol. 33, pp. 1360-1367, 2012.

[21] S. G. Advani and C. L. Tucker, "The Use of Tensors to Describe and Predict Fiber Orientation in Short Fiber Composites," *Journal of Rheology,* vol. 31, pp. 751-784, Nov 1987.

[22] S. G. Advani and C. L. Tucker, "Closure Approximations for 3-Dimensional Structure Tensors," *Journal of Rheology,* vol. 34, pp. 367-386, Apr 1990.

[23] D. H. Chung and T. H. Kwon, "Fiber orientation in the processing of polymer composites," *Korea-Australia Rheology Journal,* vol. 14, pp. 175-188, Dec 2002.

[24] D. H. Chung and T. H. Kwon, "Invariant-based optimal fitting closure approximation for the numerical prediction of flow-induced fiber orientation," *Journal of Rheology,* vol. 46, pp. 169-194, Jan-Feb 2002.

[25] H. M. Huynh, "Improved Fiber Orientation Predictions for Injection-Molded Composites," M.S., Mechanical Engineering, University of Illinois at Urbana-Champaign, 2000.

[26] M. Doi and S. F. Edwards, *The theory of polymer dynamics*. Oxford: Clarendon Press, 1986.

[27] B. N. Nguyen, S. K. Bapanapalli, J. D. Holbery, M. T. Smith, V. Kunc, B. J. Frame*, et al.*, "Fiber Length and Orientation in Long-Fiber Injection-Molded Thermoplastics – Part I: Modeling of Microstructure and Elastic Properties," *Journal of Composite Materials,* vol. 42, pp. 1003-27, 2008.

[28] W.-K. Chin, H.-T. Liu, and Y.-D. Lee, "Effects of Fiber Length and Orientation Distribution on the Elastic Modulus of Short Fiber Reinforced Thermoplastics," *Polymer Composites,* vol. 9, pp. 27-35, February 1988 1988.

[29] G. M. Vélez-García, P. Wapperom, V. Kunc, D. G. Baird, and A. Zink-Sharp, "Sample preparation and image acquisition using optical-reflective microscopy in the measurement of fiber orientation in thermoplastic composites," *J Microscopy-Oxford.,* Accepted (2012).

[30] G. M. Vélez-García, P. Wapperom, D. G. Baird, A. O. Aning, and V. Kunc, "Unambiguous orientation in short fiber composites over small sampling area in a center-gated disk," *Composites Part A: Applied Science and Manufacturing,* vol. 43, pp. 104-113, 2012.

[31] J. T. Hofmann, G. M. Velez-Garcia, D. G. Baird, and A. R. Whittington, "Application and Evaluation of the Method of Ellipses for Measuring the Orientation of Long, Semi-Flexible Fibers," *J Composites, Part A (Submitted),* 2012.

[32] C. L. Tucker, "Flow Regimes for Fiber Suspensions in Narrow Gaps," *Journal of Non-Newtonian Fluid Mechanics,* vol. 39, pp. 239-268, May 1991.

[33] J. Wang, C. A. Silva, J. C. Viana, F. W. J. van Hattum, A. M. Cunha, and C. L. Tucker, "Prediction of fiber orientation in a rotating compressing and expanding mold," *Polymer Engineering and Science,* vol. 48, pp. 1405-1413, Jul 2008.

[34] B. E. VerWeyst, C. L. Tucker, and P. H. Foss, "The optimized quasi-planar approximation for predicting fiber orientation in injection-molded composites," *International Polymer Processing,* vol. 12, pp. 238-248, Sep 1997.

[35] B. E. VerWeyst and C. L. Tucker, "Fiber suspensions in complex geometries: Flow/orientation coupling," *Canadian Journal of Chemical Engineering,* vol. 80, pp. 1093-1106, Dec 2002.

[36] G. G. Lipscomb, M. M. Denn, D. U. Hur, and D. V. Boger, "The Flow of Fiber Suspensions in Complex Geometries," *Journal of Non-Newtonian Fluid Mechanics,* vol. 26, pp. 297-325, Jan 1988.

[37] S. M. Dinh and R. C. Armstrong, "A Rheological Equation of State for Semiconcentrated Fiber Suspensions," *Journal of Rheology,* vol. 28, pp. 207-227, 1984.

[38] B. N. Nguyen, S. K. Bapanapalli, J. D. Holbery, M. T. Smith, V. Kunc, B. J. Frame*, et al.*, "Fiber length and orientation in long-fiber injection-molded thermoplastics - Part I: Modeling of microstructure and elastic properties," *Journal of Composite Materials,* vol. 42, pp. 1003-1029, May 2008.

[39] D. G. B. K. C. Ortman, P. Wapperom, A. Aning, "Prediction of Fiber Orientation in the Injection Molding of Long Fiber Suspensions," *Journal of Composite Materials,* Accepted.

[40] J. T. Hofmann, G. M. Velez-Garcia, D. G. Baird, and A. Whittington, "Application and evaluation of the method of ellipses for measuring the orientation of long, semi-flexible fibers," *Polymer Composites,* vol. 34, pp. 390-398, 2013.

3.10 Figures

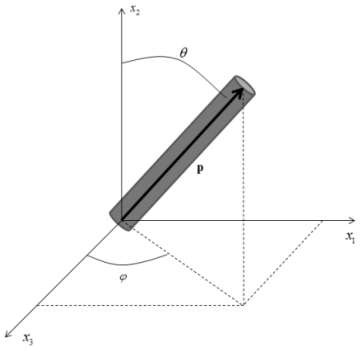

Figure 3.1: Vector definition of rigid-fiber. The fiber is characterized by the vector, $\overline{p}$, as well as the azimuthal and zenith angles, ϕ and θ respectively.

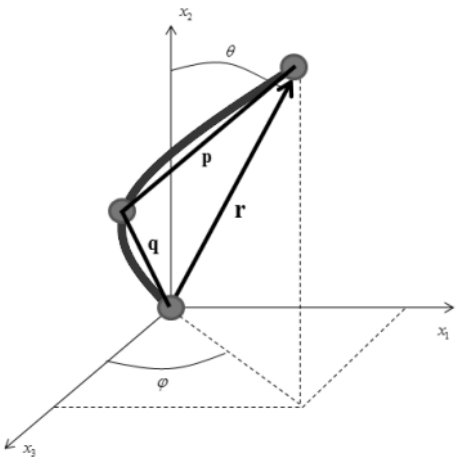

Figure 3.2: Vector definition of flexible fiber. The fiber is characterized by the vectors, $\overline{p}$ and $\overline{r}$, as well as the azimuthal and zenith angles, ϕ and θ respectively. The two vector are of equal length, l_b, and there is an internal resistivity to bending, k. The end-to-end vector, $\overline{r}$,is also defined to determine a fiber's "average" direction.

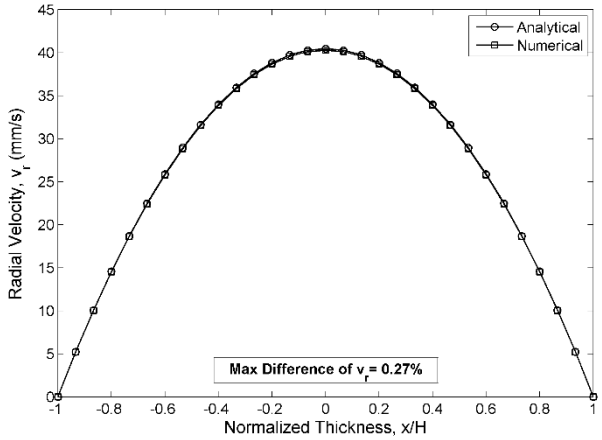

Figure 3.3: Comparison of numerical (o) and analytical (□) solution for radially diverging flow using the Hele-Shaw approximation at 40% of the mold fill. Maximum difference is 0.27%.

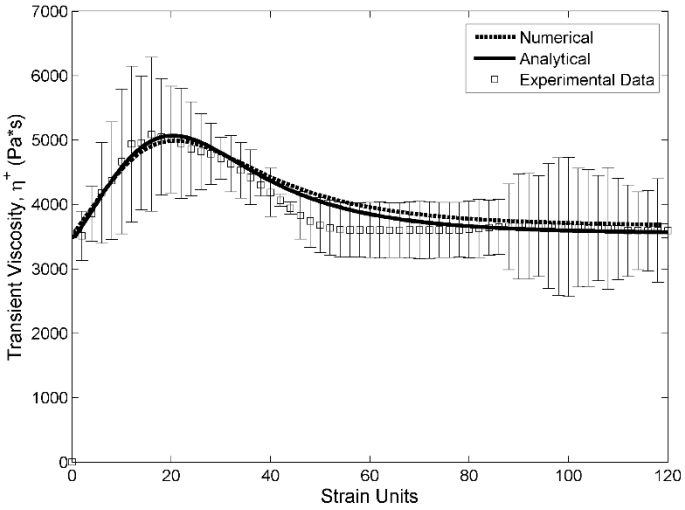

Figure 3.4: Comparison of numerical (dashed) and analytical (solid) solution of transient viscosity using the semi-flexible fiber model in simple shear flow using a sliding plate rheometer ($T = 180°C, \alpha = 0.13, C_I = 0.04, k = 218s^{-1}$, $\phi = 0.1447, l_b = 1.9mm$ $c_1 = 49, c_2 = 25000$).

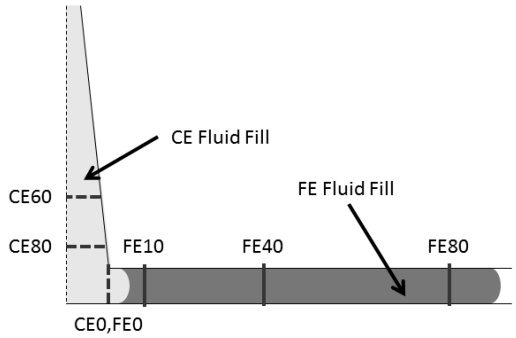

Figure 3.5: Qualitative figure to examine initial conditions entering the mold cavity. The lightly shaded area represents the currently evolving (CE) data while the combination of the lightly shaded and darkly shaded areas constitutes the fully evolved (FE) data. Areas of interested for the CE data include 60% of sprue (CE60), 80% of the sprue (CE80), and the 0% fill of the mold (CE0). Areas of interest for the FE data include 0% mold fill (FE0) 10% mold fill (FE10), 40% mold fill (FE40), and 80% mold fill (FE80).

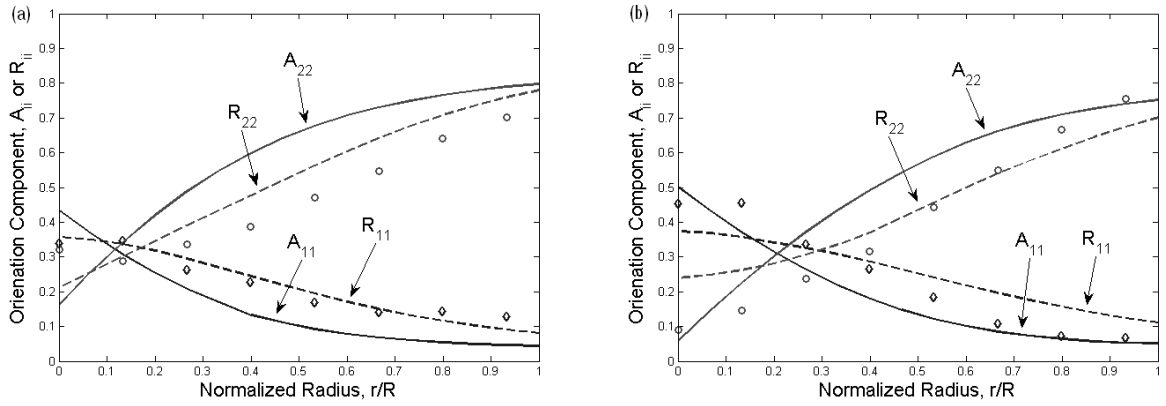

Figure 3.6: Comparing the rigid fiber (solid) model and semi-flexible (dashed) model to experimental LGF data for A_{11} (◊) and A_{22} (○) obtained from center-gated sprue when the fluid has just turned to enter the mold cavity. Comparison of models for (a) CE60 data and (b) CE80 data region of interest.

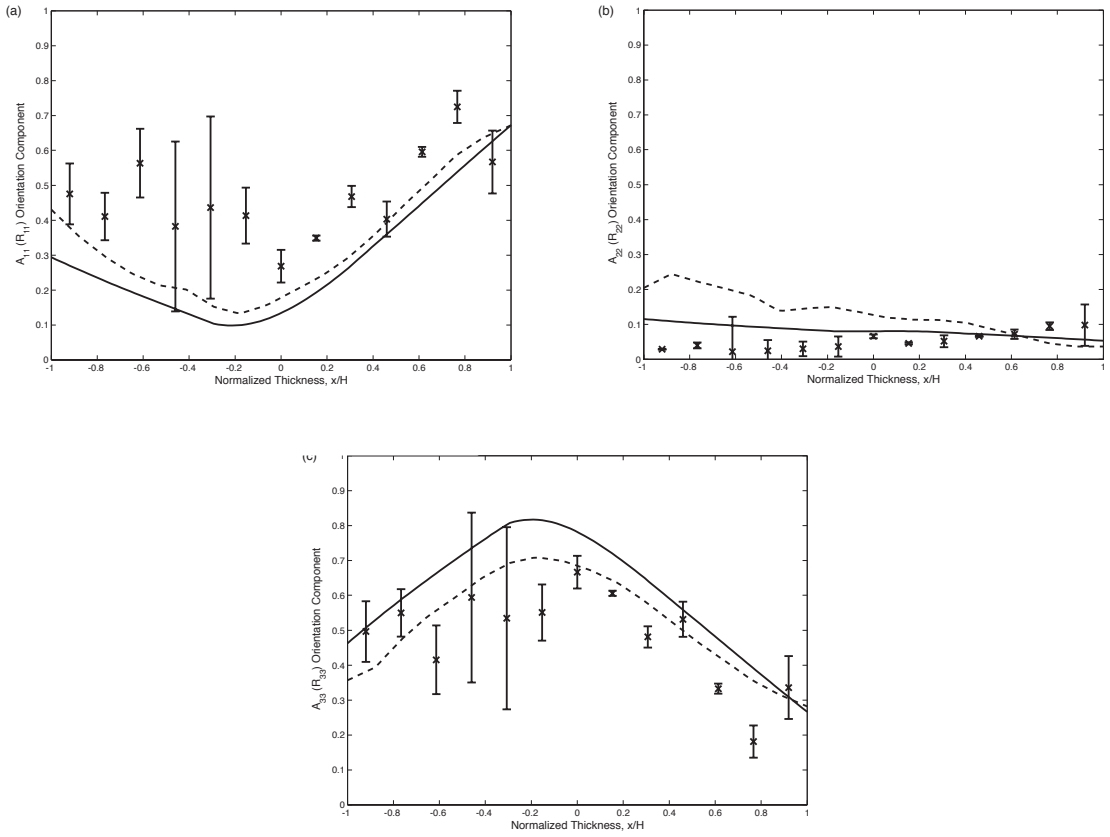

Figure 3.7: Comparison of rigid fiber (solid) and semi-flexible (dashed) model predictions versus the experimentally obtained CE0 (a) A_{11} or R_{11}, (b) A_{22} or R_{22} and (c) A_{33} or R_{33}.

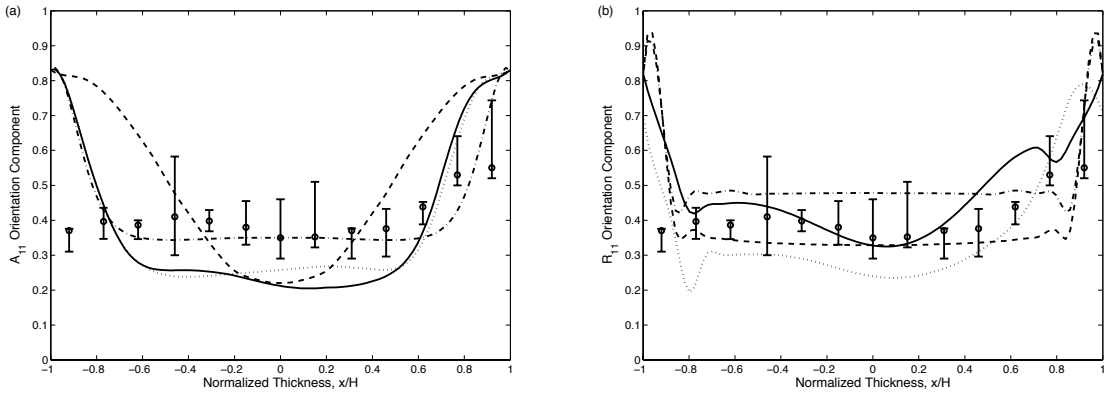

Figure 3.8: Results from simulating the center-gated disk filling operation taken FE10 of the mold fill for LGF. (a) A_{11} orientation component results using the rigid fiber rigid fiber model and (b) R_{11} orientation component results using the semi-flexible fiber semi-flexible model. Experimental IC's (dotted), random IC's (dashed), planar random IC's (dash-dot), and simulated IC's (solid) compared to experimental data (◊).

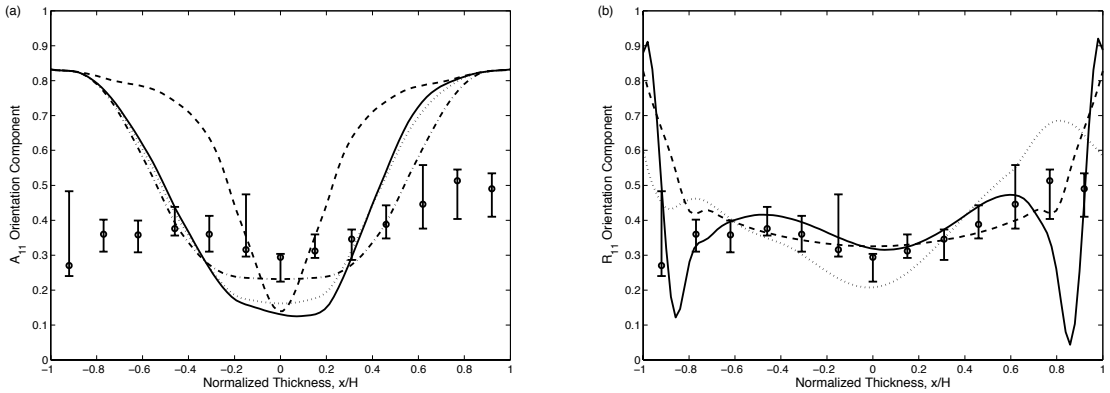

Figure 3.9: Results from simulating the center-gated disk filling operation taken at FE40 of the mold fill for LGF. (a) A_{11} orientation component results using the rigid fiber rigid fiber model and (b) R_{11} orientation component results using the semi-flexible fiber semi-flexible model. Experimental IC's (dotted), random IC's (dashed), planar random IC's (dash-dot), and simulated IC's (solid) compared to experimental data (◊).

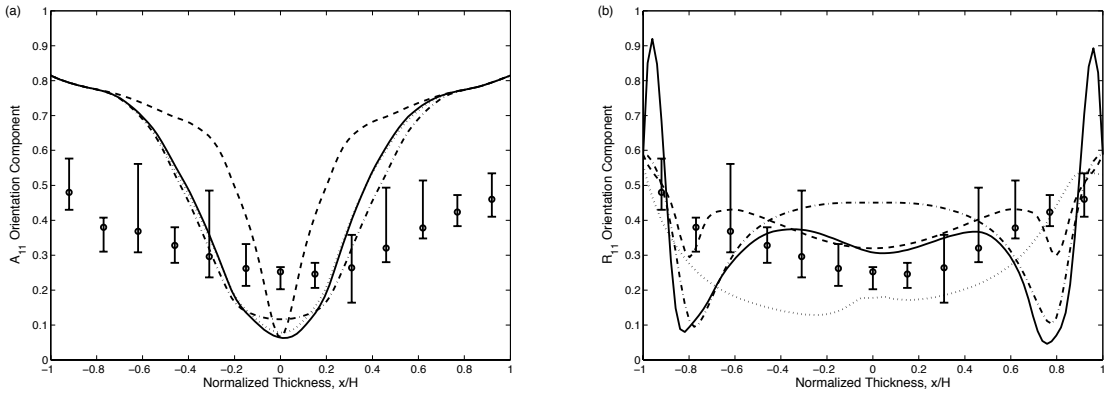

Figure 3.10: Results from simulating the center-gated disk filling operation taken at FE80 of the mold fill for LGF. (a) A_{11} orientation component results using the rigid fiber rigid fiber model and (b) R_{11} orientation component results using the semi-flexible fiber semi-flexible model. Experimental IC's (dotted), random IC's (dashed), planar random IC's (dash-dot), and simulated IC's (solid) compared to experimental data (◊).

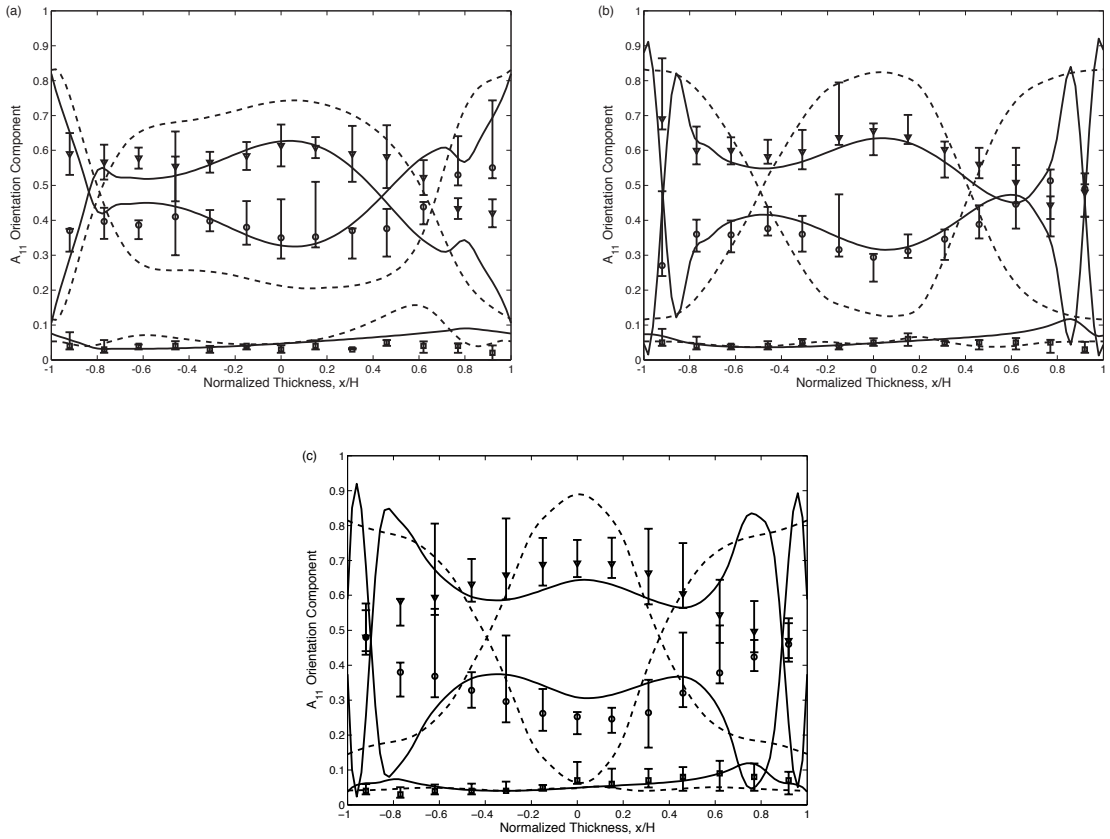

Figure 3.11: Results from simulating the center-gated disk filling operation taken at (a) FE10 of the mold fill, (b) FE40 of the mold fill, and (c) FE80 of the mold fill. Simulations are carried out using S-G-M method for long glass fibers. Predictions are made with the rigid fiber model (solid) and flexible fiber model (dash-dot) for $A_{11}\left(R_{11}\right)$ (○), $A_{22}\left(R_{22}\right)$ (□), and $A_{33}\left(R_{33}\right)(\nabla)$ orientation components.

η_0 (Pa·s)	582.14
λ (s^{-1})	0.2531
n	0.7597

Table 3.1: Parameters for the filling of polypropylene matrix filling the center-gated mold.

Chapter 4

Prediction of Short Glass Fiber Orientation in the Filling of an End-Gated Plaque

Preface

This chapter describes the prediction of short glass fiber orientation in the filling of a three-dimensional end-gated plaque using two rigid fiber models and comparing to experimentally determined values of orientation. This chapter is organized as a manuscript for publication in Composites Part A: Applied Science and Manufacturing.

4 Prediction of Short Glass Fiber Orientation in the Filling of an End-Gated Plaque

Kevin J. Meyer[1], John T. Hofmann[2], and Donald G. Baird[1]*

1. Department of Chemical Engineering, Virginia Tech, Blacksburg, VA 24061
2. Macromolecule and Interfaces Institute, Virginia Tech, Blacksburg, VA 24061

* Corresponding Author: Tel. +1 540 231 5998 ; Fax +1 540 231 2732.
Email Address: dbaird@vt.edu (D.G. Baird)
Present Address:
Department of Chemical Engineering
133 Randolph Hall
Virginia Tech

Blacksburg, VA 24061

4.1 Abstract

This work is concerned with predicting short $(L \leq 1\text{mm})$ glass fiber (SGF) orientation generated during the filling of an end-gated plaque (EGP). Previous EGP simulations have provided results only within the mold cavity and only along the centerline of the mold and made assumptions about fiber orientation at the mold entry. This paper reports on a method to simulate the flow in the sprue, gate and mold region (SGM) to obtain fiber orientation predictions within the end-gated plaque using orientation parameters fit to experimental data. Predicted values of orientation are compared to experimental data both along and away from the centerline. It is observed that orientation can be accurately predicted in a three dimensional mold cavity using the strain reduction factor model. Furthermore, initial conditions at the entrance to mold cavity appear to be a function of mold width as well as mold thickness.

4.2 Introduction

Short glass fiber composite suspensions are typically processed by injection or compression molding to form a part with a complex layered microstructure [1]. Local variations of fiber orientation are often seen in even the simplest molding geometries and have led to a variation of the properties of the final part [2]. As a result, a large amount of effort has focused on predicting the final orientation of fibers in an injection or compression molded part to optimize the processing conditions and part performance.

Modeling the orientation of concentrated short glass fiber (SGF) systems during mold filling has typically been accomplished through some variant of Jeffery's model [3]. Jeffery described the motion of a single prolate spheroid in a Newtonian fluid in creeping flow. Folgar and Tucker [4] modified Jeffery's model by adding a term based on isotropic rotary diffusivity that was proportional to the velocity gradient to account for fiber-fiber interaction in a concentrated fiber system and weighted by an empirical value, C_I. The fiber interaction coefficient, C_I, has been estimated by fitting experimental data [5], through empirical relationships [6, 7] and by fitting

transient stress growth at the startup of steady shear [8, 9]. The Folgar-Tucker (FT) model has shown good qualitative agreement with experimental data and is thus a popular choice for simulating short glass fiber orientation [5, 10].

Stress growth experiments in the startup of simple shear have shown that the orientation of fibers in concentrated suspension evolves more slowly than predicted by the Folgar-Tucker model [11, 12]. To more accurately reflect the observed transient fiber orientation a "slip" parameter was suggested by Sepehr et al. [13, 14] and Eberle et al. [11] to slow the evolution of orientation. The strain reduction factor (SRF) model proved more accurate in comparison to experimental data but the addition of the slip parameter rendered the SRF model non-objective [15]. Wang et al. [16] developed the reduced strain closure (RSC) model as an objective form of the SRF model where the closure approximation is modified eliminating the objectivity problem while including the slower orientation kinetics observed in experimental values of orientation. Even though the SRF model is non-objective it has been shown to be useful in describing the evolution of orientation in simple flows and in more general flows [14, 17, 18]. Furthermore, Wang et al. [19] showed that the SRF model qualitatively predicted fiber orientation in a rotating compressing and expanding center-gated disk. Phelps and Tucker [20] have developed a form of the RSC model (ARD-RSC) which accounts for the anisotropy in fiber interactions but requires additional fitting efforts in determining six model coefficients.

The prediction of short fiber orientation in complex test geometries for injection molding has been the subject of significant work. A number of authors have investigated the prediction of SGF orientation in a center-gated disk geometry and have found reasonable agreement with experimentally obtained values of fiber orientation under different conditions [5, 10, 21-24]. The prediction of SGF orientation in an end-gated plaque geometry has seen considerably less attention. Altan et al. [25] assumed a planar random inlet condition to a rectangular channel and computed the short fiber orientation using a coupled approach involving both the 4^{th} and 6^{th} order tensors of orientation but did not compare his results to experimental orientation values. Bay and Tucker [5], Han and Im [26] and Nguyen et al. [27] simulated the orientation of glass fibers in an end-gated strip and found reasonable agreement with experimental orientation values but only made comparisons along the centerline of the mold. Gupta and Wang [28] performed a more detailed analysis of an end-gated plaque using a polyester matrix and SGF's and saw reasonable qualitative agreement between predicted values of orientation and experimentally obtained values but no quantitative statement could be made due to course sampling techniques. Thus, there is a need for a more comprehensive and quantitative study of predicting the orientation of SGF's in an end-gated plaque geometry.

Predicting fiber orientation in complex geometries has been shown to be affected by the choice of initial conditions. Previous authors have observed that assumed symmetric initial conditions for fiber orientation produce symmetric orientation predictions [5, 23, 24]. In contrast, experimentally obtained values for fiber orientation are generally observed to be asymmetric through the mold thickness and thus require asymmetric initial conditions [24]. One method of obtaining asymmetric initial conditions of orientation involves the use of experimental orientation values obtained from the gate mold interface. Velez-Garcia et al. [23] used experimental data as initial conditions for the simulation of fiber orientation to the mold in a center-gated disk and found an increased agreement with experimental orientation values when compared to an assumed orientation. The drawback to this method is that it requires experimental knowledge of the system that is being predicted (to predict the orientation of fibers

in a system experimental data must first be obtained and analyzed from that system). To address this concern, Chung and Kwon [24] suggested simulating the entire mold of a center-gated disk (sprue and mold) as a single domain and saw a similar increase in agreement between predicted values of orientation and experimental data for a SGF system as Velez-Garcia [23] but did not require experimental orientation values. Meyer et al. [21] built on the work of Chung and Kwon [24] and developed the sprue-gate-mold (SGM) method for the prediction of orientation of long glass fiber suspensions in the center-gated disk and observed an increase in agreement between model predictions and experimental data again without the need for experimental orientation values at the gate/mold interface.

The purpose of this work is to predict short glass fiber orientation in an end-gated plaque including gate effects, the advancing front and temperature effects and compare the results with experimentally obtained values of short glass fiber orientation both along and away from the centerline of the mold drawing conclusions as to the accuracy of the predictions. Furthermore, it will be determined if objectivity plays a role in predicting SGF orientation throughout the end-gated plaque by solving both the SRF and RSC models throughout the geometry. The fiber orientation predictions are performed using the decoupled approach for the stress and orientation tensors [22, 29, 30]. A volume of fluid finite element method is used in the ANSYS Polyflow environment to calculate the velocity gradients of a non-isothermal generalized Newtonian fluid filling an end-gated plaque geometry incorporating both the gate region and advancing front. The solutions of the SRF and RSC fiber orientation models are calculated through a combination of MATLAB and C routines in the MATLAB environment. Fiber orientation predictions are carried out using orientation parameters (κ and C_I) obtained through fitting transient stress data from shear flow experiments and through fitting experimental data obtained from injection molding experiments.

4.3 Governing Equations

4.3.1 Flow and Heat Equations

The flow was assumed laminar $\left(\mathrm{Re} \approx 10^{-3}\right)$ and incompressible resulting in the equations of continuity and motion as given in Eq. (4.1) and Eq. (4.2), respectfully:

$$0 = \nabla \cdot \overline{v} \tag{4.1}$$

$$0 = -\nabla P + \nabla \cdot \overline{\overline{\tau}} \tag{4.2}$$

In Eqs. (4.1) and (4.2), $\overline{v}$ is the velocity vector, P is the isotropic pressure and $\overline{\overline{\tau}}$ is the extra stress tensor.

Temperature was also taken into account in the simulations using Eq. (4.3) where ρ is the density, $\overline{C}_P$ is the specific heat capacity per unit mass, DT/Dt is the material derivative $\left(\partial T/\partial t + \overline{v} \cdot \nabla T\right)$, k is the thermal conductivity, $\overline{\overline{\dot{\gamma}}}$ is the rate of strain tensor and $\overline{\overline{\tau}}$ is the extra stress tensor:

$$\rho \overline{C}_P \frac{DT}{Dt} = k\nabla T + \overline{\overline{\tau}} : \overline{\overline{\dot{\gamma}}} \tag{4.3}$$

The viscosity of the matrix was also dependent on the temperature of the system and is included through the use of an Arrhenius law relation shown in Eq. (4.4) where $\alpha = E_a/R$ and T_α is a reference temperature:

$$\eta(T) = \exp\left[\alpha\left(\frac{1}{T} - \frac{1}{T_\alpha}\right)\right] \tag{4.4}$$

For the duration of the work presented here the fluid is assumed to have constant thermal conductivity and heat capacity. The parameters for Eq. (4.3) and Eq. (4.4) are given in Table 4.1.

4.3.2 Extra Stress Tensor Representation

The decoupled approach for the solution of the stress and orientation tensors was used based on previous work by Chung and Kwon [24], Eberle et al. [8], Ortman et al. [9] and Mazahir et al. [22] which suggests there is no consistent representation of the stress tensors which includes both fiber loading and fiber-fiber interaction for concentrated suspensions. From the perspective of the fibers, the major drag experienced is from the polymer melt. The effect of fibers was included by incorporating the fiber loading and fiber-fiber interaction contributions into the simulations through the values of κ and C_I so that only the neat matrix properties are used to predict the velocity fields. A generalized Newtonian fluid was chosen as the constitutive relation, given in Eq. (4.5), to represent the matrix rheology used in this study where $\eta(\dot{\gamma})$ is an empirical relation for the viscosity and $\overline{\overline{\dot{\gamma}}} = \frac{1}{2}\left[\nabla\overline{v} + \nabla\overline{v}^T\right]$ where $\nabla\overline{v} = \partial v_j/\partial x_i$:

$$\overline{\overline{\tau}} = \eta(\dot{\gamma})\overline{\overline{\dot{\gamma}}} \tag{4.5}$$

The Cross model was used to account for the shear-thinning nature of neat matrix and is given by Eq. (4.6) where η_0 is the zero shear viscosity, λ is the parameter which governs the onset of shear thinning and n is a parameter which governs the degree of shear thinning within the matrix:

$$\eta(\dot{\gamma}) = \frac{\eta_0}{1 + (\lambda\dot{\gamma})^{1-n}} \tag{4.6}$$

The parameters for the Cross model can be found in Table 4.1 and were obtained through testing in a rotational rheometer (RMS-800, Rheometrics, Inc.).

4.3.3 Fiber Orientation Equations

A convenient way of representing the orientation of fibers within an injection-molded part is through the use of a 2nd order orientation tensor (shown in Eq. (4.7)) given by Advani and Tucker [31] where $\overline{\overline{A}}$ is the orientation tensor, $\overline{p}$ is a vector running through the longitudinal axis of the individual fiber (shown in Figure 4.1) and $\psi(\overline{p},t)$ is the probability distribution function:

$$\overline{\overline{A}} = \iint \overline{pp}\psi(\overline{p},t)d\overline{p} \tag{4.7}$$

In the equations of orientation, a 4th order tensor also arises due to the drag of the fluid on the fiber and is given in Eq. (4.8):

$$\overline{\overline{A}}_4 = \iint \overline{pppp}\psi(\overline{p},t)d\overline{p} \tag{4.8}$$

The 4th order orientation tensor in Eq. (4.8) requires the use of a closure approximation and a summary of many popular closure approximations is given by Chung and Kwon [32]. [32]. Recent work in the area of closure approximations has yielded the Fast Exact closure and Neural Network-based closures [33, 34]. For the purpose of this work, the invariant-based optimal fitted closure (IBOF) is used which is based on the 2nd and 3rd invariants of the 2nd order orientation tensor, $\overline{\overline{A}}$ [35, 36]. Furthermore, the IBOF closure has been shown to produce solutions very close to direct calculation of the probability distribution function and is very computationally efficient in comparison to other orthotropic closures [36].

The strain reduction factor (SRF) model is given in Eq. (4.9) where $\overline{\overline{A}}$ is the second moment of the orientation distribution function, $\dot{\overline{\overline{A}}}$ is the material derivative of $\overline{\overline{A}}$ $\left(\dot{\overline{\overline{A}}} = \partial\overline{\overline{A}}/\partial t + \overline{v}\cdot\nabla\overline{\overline{A}}\right)$, $\overline{\overline{A}}_4$ is the fourth moment of the orientation distribution function, $\overline{\overline{W}}$ is the vorticity tensor $\left(\overline{\overline{W}} = \frac{1}{2}\left[\nabla\overline{v} - \nabla\overline{v}^T\right]\right)$, $\overline{\overline{D}}$ is the rate of deformation tensor $\left(\overline{\overline{D}} = \frac{1}{2}\left[\nabla\overline{v} + \nabla\overline{v}^T\right]\right)$, $\overline{\overline{I}}$ is the unit tensor, ξ is a shape factor (usually given a value of unity for fiber systems), $\dot{\gamma}$ is the magnitude of the rate of deformation tensor $\left(\dot{\gamma} = \sqrt{\frac{1}{2}\left[\overline{\overline{D}} : \overline{\overline{D}}\right]}\right)$ and κ and C_I are phenomenological constants based on the suspension properties [4, 13]:

$$\dot{\overline{\overline{A}}} = \kappa\left[\left(\overline{\overline{W}}\cdot\overline{\overline{A}} - \overline{\overline{A}}\cdot\overline{\overline{W}}\right) + \xi\left(\overline{\overline{D}}\cdot\overline{\overline{A}} + \overline{\overline{A}}\cdot\overline{\overline{D}} - 2\overline{\overline{A}}_4 : \overline{\overline{D}}\right) + 2\dot{\gamma}C_I\left(\overline{\overline{I}} - 3\overline{\overline{A}}\right)\right] \tag{4.9}$$

The orientation equation in Eq. (4.9) has seen wide use since it's introduction by Sepehr et al. [14] and because the vorticity tensor, $\overline{\overline{W}}$, is multiplied by the strain reduction factor, κ, the equation in not objective [15, 21, 23] .

The reduced strain closure (RSC) model was introduced as the objective form of Eq. (4.9) and is given in Eq. (4.10) where the reduction factor, κ, has been moved into the closure approximation so that it only effects the objective tensors [16]:

$$\begin{aligned}\dot{\overline{\overline{A}}} = \overline{\overline{W}}\cdot\overline{\overline{A}} - \overline{\overline{A}}\cdot\overline{\overline{W}} + \xi\left\{\overline{\overline{D}}\cdot\overline{\overline{A}} + \overline{\overline{A}}\cdot\overline{\overline{D}} - 2\left[\overline{\overline{A}}_4 + (1-\kappa)\left(\overline{\overline{L}}_4 - \overline{\overline{M}}_4 : \overline{\overline{A}}_4\right)\right] : \overline{\overline{D}}\right\} \\ + 2\kappa C_I\dot{\gamma}\left(\overline{\overline{I}} - 3\overline{\overline{A}}\right)\end{aligned} \tag{4.10}$$

The use of the RSC model requires two 4th order tensors, $\overline{\overline{L}}_4$ and $\overline{\overline{M}}_4$, which are based on the eigenvalues (λ_i) and eigenvectors $(\overline{e}_i)$ of the 2nd order orientation tensor, $\overline{\overline{A}}$, shown in Eq. (4.11) and Eq. (4.12):

$$\overline{\overline{L}}_4 = \sum_{i=1}^{3} \lambda_i \overline{e}_i \overline{e}_i \overline{e}_i \overline{e}_i \tag{4.11}$$

$$\overline{\overline{M}}_4 = \sum_{i=1}^{3} \overline{e}_i \overline{e}_i \overline{e}_i \overline{e}_i \tag{4.12}$$

Obtaining the value of the strain reduction factor, κ, and the fiber interaction coefficient, C_I, for both the SRF and RSC models will be discussed in the results section of this paper.

4.4 Solution Method

4.4.1 Finite Element Simulations

The filling of the cavity was simulated using the ANSYS Polyflow finite element (FEM) software package. The entrance to the gate region was meshed using 31 (thickness) x 30 (length) x 20 (width) elements. The gate region was meshed using 31 (thickness) x 30 (length) x 50 (width) hexagonal elements. The mold was meshed using 31 (thickness) x 300 (length) x 50 (width) hexagonal elements. A lower number of elements caused convergence issues in the fiber orientation calculations while a higher number of elements in any of the three meshed areas saw no increase in accuracy gained. Using unequal grid spacing captured the large gradients encountered near boundaries.

The volume of fluid (VOF) method was used inside the ANSYS Polyflow environment to simulate the transient mold filling process [37]. The VOF method solves the transport equation for the volume fraction of fluid at each node where ϕ is the volume fraction of the liquid phase and $\overline{v}$ is the velocity vector at that node given by Eq. (4.13):

$$\frac{\partial \phi}{\partial t} + v \cdot \nabla \phi = 0 \tag{4.13}$$

The solution of this equation allows for the fluid front to be tracked as it moves through the empty cavity thereby incorporating the "fountain flow" effect in the simulations. The influence of this complex moving front has been shown to impact fiber orientation predictions in axisymmetric radially diverging flow [5, 10, 24].

The solution of the non-isothermal system required the use of an evolution scheme inside the ANSYS Polyflow environment to control the viscous dissipation and convection term associated with Eq. (4.3). The viscous dissipation term and convection term were scaled based on the time step so that at small time steps these terms had little effect on the final solution. The terms influence was slowly increased until 20% of the mold fill simulation was completed where they were no longer scaled and provided a stable solution.

4.4.2 Finite Difference Simulations

The fiber orientation equations were solved using 2nd order accurate finite differences (FDM) in MATLAB (The Mathworks Inc., ver. 7.4) and C on a stationary mesh. The equations used to predict fiber orientation are originally hyperbolic partial differential equations because of the velocity vector in the convective term. But, because the stress tensor and orientation tensor are decoupled in this study, the velocity vector is known so that the convective term can be explicitly written. This reduces the set of partial differential equations to non-linear ordinary

differential equations. The ordinary differential equations are solved using a variable step size backwards differences implementation of the Kopfenstein-Shampine family of numerical differentiation formulas [38]. Fiber orientation predictions in the fountain flow region are calculated through the use of backward differentiation methods.

4.4.3 Boundary Conditions

The non-isothermal FEM simulations of the end-gated plaque (shown in Figure 4.2) requires specified boundary conditions for both the flow and heat equations. The inlet boundary, given by Ω_{inlet}, required an inlet fluid temperature $\left(T_{in} = 463\text{K}\right)$ and volumetric flow rate $\left(V_{in} = 3225\,\frac{\text{mm}^3}{\text{s}}\right)$ and are based on experimental conditions. The symmetry condition $\left(\overline{f}_s = 0, \overline{v}_n = 0\right)$ is specified by Ω_{sym}. At the walls of the mold both the no slip condition $\left(\overline{v} = 0\right)$ and the mold wall temperature $\left(T_{wall} = 363\text{K}\right)$ were specified. Because the VOF method here does not calculate the movement of air out of the mold, the only other constraint was to set both the normal and tangential forces to zero at the moving fluid boundary $\left(\overline{f}_n = \overline{f}_t = 0\right)$. The FDM simulation required only one boundary condition for fiber orientation to be specified at the inlet of the gate $\left(\overline{\overline{A}} = \frac{1}{3}\overline{\overline{I}}\right)$ and did not affect orientation predictions. Fiber orientation on all other boundaries $\left(\Omega_{outlet}, \Omega_{wall}, \Omega_{sym}\right)$ is based on the velocity fields at those locations.

4.5 Experimental Data

4.5.1 Injection Molding Conditions

End-gated plaques (EGP) were formed for the experimental evaluation of fiber orientation using 30 weight percent short glass fiber in a polypropylene matrix (RTP 105, The RTP Company, Inc.). The temperatures of the feed, compression and metering zones within the injection molding machine (Arburg Allrounder, Model 221-55-250) were set to 190°C, 190°C and 190°C, respectively, while the mold temperature was held at 90°C. The sprue length for the mold was 65mm with an initial radius of 1.45mm and a final radius at the gate of 1.75mm. The gate region of the plaque has dimensions of 80.68mm (width) by 6.25mm (height) by 6.33mm (length) and the mold region of the plaque has dimensions of 75.05mm (width) by 1.55mm (height) by 77.65mm (length). The plaques were molded using a fill time of 2.00 seconds with a backpressure of 20 MPa.

4.5.2 Fiber Length Distribution

The fiber length distribution (FLD) was determined to verify that the average lengths (l) of the fibers in the end-gated plaque are indeed in the short fiber regime that is typically defined to be $l < 1\text{mm}$. Methods described by Nguyen et al. [27] were used to determine the fiber length distribution of approximately 2000 fibers. The method involves taking a representative sample of the injection-molded sample and burning off the polymer matrix leaving only the glass fibers behind. A small amount of epoxy resin was applied to the glass fiber matrix to secure a population of fibers and any fibers not secured to the epoxy were removed. The remaining fiber/epoxy system was returned to the furnace to remove all the epoxy. Finally, the remaining fibers were analyzed through optical techniques to determine the FLD. This method produced a number average length and diameter of $0.923 \pm 0.03\text{mm}$ and $13.9\,\mu\text{m}$, respectively.

4.5.3 Fiber Orientation Measurements

Samples of the injected molded composite were chosen within the end-gated mold geometry at a 0%, 10%, 40% and 90% of the length of the mold at the centerline, 50% width of the mold and 90% width of the mold (sampled locations are indicated by gray shading in Figure 4.2). The selected samples were prepared using the method prescribed by Velez-Garcia et al. [39]. Optical microscopy was used to analyze samples at each inspection line using 20X magnification and motorized stage with image stitching software (Nikon Eclipse LV100, NIS-Elements Basic Research software, v. 3.10). The resulting images had a width of $700\ \mu$m and height equal to the thickness of the disk. The unambiguous components of the orientation tensor were determined through the method of ellipses using a traditional size bin width of 0.8 mm [40, 41].

4.6 Results

4.6.1 Obtaining Fiber Orientation Parameters

The prediction of short fiber orientation uses parameters (the slip parameter κ and the fiber interaction coefficient C_I) that strongly dictate the final orientation structure through the mold cavity thickness. Initially, the orientation parameters κ and C_I were obtained through the fitting of transient stress growth data at the startup of simple shear flow as described by Eberle et al. [11] and Ortman et al. [9]. The calculated values of the two fiber orientation parameters were determined through the fitting of experimental stress and fiber orientation data at the startup of simple shear deformation at $\dot{\gamma} = 1\ \text{s}^{-1}$ with resulting values of $\kappa = 0.3436$ and $C_I = 0.0309$. These parameters were used to predict the fiber orientation in the end-gated plaque filling process with the resultant predictions shown by the solid lines in Figure 4.3. The fiber orientation predictions shown in Figure 4.3 appear to produce very distinct shell-core-shell regions that is not reflected in the experimental data suggesting that the value of the slip parameter, κ, may be too high. Furthermore, at no point within the mold filling comparisons along the centerline do the predicted values of orientation agree with the experimentally determined values. The value of the fiber interaction coefficient, C_I, appears to accurately predict fiber orientation near the walls of the mold suggesting that this value may be correct.

Due to the poor agreement between the orientation predictions using the parameters obtained from simple shear flow, experimental data at the 0% fill position in the mold along the centerline was fitted (position (1) in Figure 4.2). The fitting was performed by simulating the filling of the end-gated sprue, gate and mold and comparing the predicted values of orientation to the observed experimental values of orientation at the position of interest (position (4) in Figure 4.2) and adjusting the orientation parameters accordingly. The fitting procedure used an error reducing (error between the experimental and predicted values of the R_{11} and R_{33} orientation components) non-linear least squares (NLLSQ) analysis to fit both the SRF and RSC model predictions to the components of experimental orientation at this fill position. The values of the orientation parameters obtained from this position were then used for all other predictions in the end-gated geometry both along and away from the centerline. The results of the fitting procedure yielded values of $\kappa = 0.0297$ and $C_I = 0.0481$ for the non-objective SRF model and values of $\kappa = 0.0910$ and $C_I = 0.0658$ for the objective RSC model.

4.6.2 Comparison of Objective and Non-Objective Fiber Orientation Models

In order to assess the role objectivity plays in the prediction of short glass fiber orientation in the end-gated plaque the SRF model predictions, given by Eq. (4.9), and the RSC model predictions, given by Eq. (4.10), were compared at four locations. Figure 4.4 (a) (position (1) in Figure 4.2) shows the results of the SRF and RSC model predictions compared to experimental data at the 0% fill position along the centerline of the end-gated plaque. It is observed that both models predict similar values of orientation through the mold cavity thickness and also both accurately predict experimental values of SGF orientation at this position. The RSC and SRF models are compared at 10% fill along the centerline (position (7) in Figure 4.2) and it is observed that the predictions for both models are very similar and within or very close to the error associated with the experimentally obtained values. The third position of interest is 0% fill at 50% of the width of the end-gated mold cavity (position (2) in Figure 4.2) where the SRF and RSC models produce similar numerical predictions for SGF orientation through the mold cavity thickness with neither model performing significantly better or worse when compared to experimentally obtained orientation data. Finally, the SRF and RSC models are compared at 40% fill and 50% width of the end-gated mold cavity (position (8) in Figure 4.2). At this position well away from the centerline it is again observed that the SRF and RSC models predict very similar values of orientation through the mold cavity and it cannot be concluded that either is more accurate at predicting experimental fiber orientation within the error associated with the measurements. Additionally, the RSC model and SRF model were compared at a number of other locations and similar results were observed suggesting that objectivity may not play a large role in the prediction of SGF orientation in the end-gated plaque.

4.6.3 Fiber Orientation Predictions at 0% Mold Width

The SRF model was used to predict fiber orientation along the centerline of the mold cavity of the end-gated plaque. The SRF model is compared to experimental data at 0% of the mold fill along the centerline in

Figure 4.5 (a) (position (1) in Figure 4.2). At this position good agreement is observed between the model predictions and experimentally observed data accurately predicting a shell-core-shell region commonly associated with SGF composites in thin cavities.

Figure 4.5 (b) (position (4) in Figure 4.2) shows the results of the SRF model at 10% of the mold fill along the centerline of the end-gated plaque. At this position, the SRF model is observed to agree well with experimentally obtained SGF orientation values again accurately predicting the shell-core-shell region. The SRF model is compared to experimental SGF orientation values at 40% of the mold fill along the centerline of the mold in

Figure 4.5 (c) (position (7) in Figure 4.2). The SRF model performs encouragingly for all three of the shown orientation tensor components. The A_{33} component prediction is slightly more accurate than the A_{11} component, but both are either within the experimental error or close to the values including experimental error with the exception of the center of the mold. At the center of the mold the over prediction of the A_{22} component leads to an under prediction of both the A_{11} and A_{33} components. Finally, the SRF model is compared to experimental orientation data at 90% of the mold fill along the centerline of the mold data in

Figure 4.5 (d) (position (10) in Figure 4.2). Here the experimentally obtained orientation data show a wider orientation distribution than at 40% of the mold fill. This wider distribution is

captured using the SRF model as the agreement with experimental data is encouraging. The exception is near the center of the mold where both the A_{11} and A_{33} components of orientation are slightly under predicted because of an over predicted A_{22} component of orientation. This over prediction can partially be attributed to the non-uniform velocity gradients experienced across the length of the glass fiber.

4.6.4 Fiber Orientation Predictions at 50% Mold Width

Comparisons between observed experimental fiber orientation and predicted values of orientation using the SRF model were made at 50% of the mold width. Figure 4.6 (a) shows the predicted values of fiber orientation using the SRF model and the experimentally observed orientation values at 0% mold fill and 50% width of the mold (position (2) in Figure 4.2). The SRF model is observed to perform well in predicting experimental fiber orientation at this position in the mold cavity for all three shown orientation components. Furthermore, both the predicted values and experimentally obtained values are much different than those observed at the centerline of the mold cavity. Model predictions are compared to experimental data at 10% of the mold fill at 50% percent of the mold within Figure 4.6 (b) (position (5) in Figure 4.2). Qualitatively, the shell-core-shell region observed in experimental data along the centerline of the mold is lost at this position leading to a much broader and flatter orientation distribution profile through the cavity thickness. The SRF model is observed to quantitatively capture this phenomenon for all three shown orientation tensor components and generally agrees with experimental orientation data except very near the walls. At the walls, A_{22} is over predicted leading to over predicted values of A_{11} component and an under predicted A_{33} component at the top of the mold cavity. The SRF model predictions were compared to experimentally obtained fiber orientation data at 40% of the mold fill and 50% mold width in Figure 4.6 (c) (position (8) in Figure 4.2). Here the SRF models predict a very broad fiber orientation distribution through the mold cavity thickness and is observed to agree well with experimental data for all three shown orientation components. Very near the bottom mold wall the A_{11} orientation is over predicted causing the A_{33} component to be under predicted. Finally, the SRF model was compared to experimentally obtained fiber orientation values at 90% of the mold fill at 50% of the width given in Figure 4.6 (d) (position (11) in Figure 4.2). At this position of the mold a shell-core-shell region again emerges in the experimental orientation data and is captured by the orientation predictions of the SRF model. Near the top mold wall the accuracy of the SRF model is poorer quantitatively over predicting the A_{11} component and under predicting the A_{33} component.

4.6.5 Fiber Orientation Predictions at 90% Mold Width

Orientation predictions were compared to experimentally obtained SGF orientation data at 90% of the mold width. The results in Figure 4.7 (a) show the SRF model predictions compared to experimental data at 0% mold fill and 90% mold width (position (3) in Figure 4.2). Qualitatively the SRF model captured the orientation distribution through the thickness of the cavity well but quantitatively under predicts the A_{11} component and over predicts the A_{33} component through the entire mold thickness except near the bottom wall. The results in Figure 4.7 (b) show experimental orientation data compared to predicted values of orientation using the SRF model at 10% mold fill and 90% mold width (position (6) in Figure 4.2). At position (9) the SRF model predicted SGF orientation values similar to those obtained experimentally through the mold thickness with the exception of the top mold wall where the predictions are slightly off.

Figure 4.7 (c) shows the results of the SRF model predictions to experimentally observed orientation values at 40% mold fill and 90% mold width (position (9) in Figure 4.2). Qualitatively the SRF model predicts the correct wide distribution of orientation but quantitatively the model performs poorly when compared to experimental orientation values. Finally, Figure 4.7 (d) shows the comparison of predicted orientation values using the SRF model and the values obtained from experimentation at 90% mold fill and 90% mold width (position (12) in Figure 4.2). Again the SRF model qualitatively predicts the correct wide distribution through the mold cavity thickness. Quantitatively the predicted A_{11} orientation component agrees well with experimental data in the top half of the mold while the predicted A_{33} orientation component agrees well with experimental data in the bottom half of the mold.

4.7 Conclusions

Comparisons of the model predictions and experimentally obtained values of SGF orientation were made for the non-isothermal filling of an end-gated plaque using the decoupled approach for the stress and orientation tensors incorporating the gate region and the advancing front. Fiber orientation parameters obtained through fitting transient stress overshoots in shear flow (for both the SRF and RSC models) over predicted the degree of orientation through the mold cavity thickness yielding values of orientation that suggested a stronger shell-core-shell region than was observed experimentally. This implies that the study of suspensions in simple shear alone to obtain orientation parameters may not accurately capture all phenomena associated with this system and other rheological studies may be needed (extensional flow, etc.). Due to the poor orientation predictions from the parameters fit in simple shear flow, fiber orientation parameters were obtained through the fitting of experimental data at one position in the mold (0% fill, centerline) using a non-linear least squares analysis. Using the orientation parameters determined through fitting experimental data (at 0% of the mold fill along the centerline) proved to be successful at predicting experimentally obtained values of orientation for glass fiber/polypropylene system presented here. Along the centerline, where most work has been done previously, the SRF model quantitatively predicts the experimental values of orientation at all of the locations of interest. At two positions along the centerline the A_{22} component of orientation was over predicted when compared to experimental data. It should be pointed out that anisotropic models can partially correct this issue but it comes at a much higher computational cost.

When sampling regions away from the centerline the SRF model does an encouraging job at predicting fiber orientation in all of the locations of interest verifying that the model can predict fiber orientation away from the centerline of a three-dimensional molding geometry. Additionally, at the gate/mold interface away from the centerline of the mold, fiber orientation is observed to be a function of the width and thickness of the mold cavity suggesting that general assumptions about initial conditions (fibers enter the mold cavity planar random, etc.) cannot be made here when predicting SGF orientation. Finally, the non-objective SRF model and the objective RSC model appear to exhibit similar trends at the locations sampled in the end-gated plaque suggesting that objectivity may not play a significant role in the prediction of fiber orientation in a more complex test geometry such as the end-gated plaque.

4.8 Acknowledgments

The financial support for this work from the National Science Foundation Grant No. CMMI-0853537 is gratefully appreciated and acknowledged. The authors also wish to thank the

RTP Company Inc. for supplying the neat matrix resin (RTP 100) and the 30 wt. % SGF fiber material (RTP 105). The authors also acknowledge Syed Mazahir and Mark Cieslinski for the fruitful discussions.

4.9 References

[1] S. G. Advani and E. M. Sozer, *Process modeling in composites manufacturing*, 2nd ed. Boca Raton, FL: CRC Press, 2011.

[2] M. W. Darlington and A. C. Smith, "Some features of the injection molding of short fiber reinforced thermoplastics in center sprue-gated cavities," *Polymer Composites,* vol. 8, pp. 16-21, 1987.

[3] G. B. Jeffery, "The Motion of Ellipsoidal Particles Immersed in a Viscous Fluid," *Proceedings of the Royal Society a-Mathematical Physical and Engineering Sciences,* vol. 100, pp. 161-179, 1922.

[4] F. Folgar and C. L. Tucker, "Orientation Behavior of Rigid Fibers in Concentrated Suspensions," *Journal of Rheology,* vol. 26, pp. 604-604, 1982.

[5] R. S. Bay and C. L. Tucker, "Fiber Orientation in Simple Injection Moldings .2. Experimental Results," *Polymer Composites,* vol. 13, pp. 332-341, Aug 1992.

[6] R. S. Bay, "Fiber orientation in injection molded composites: a comparison of theory and experiment," Ph.D. , University of Illinois, Urbana-Champaign, 1991.

[7] N. Phan-Thien, X. J. Fan, R. I. Tanner, and R. Zheng, "Folgar-Tucker constant for a fibre suspension in a Newtonian fluid," *Journal of Non-Newtonian Fluid Mechanics,* vol. 103, pp. 251-260, Mar 25 2002.

[8] D. G. Baird, A. P. R. Eberle, P. Wapperom, and G. M. Velez-Garcia, "Using transient shear rheology to determine material parameters in fiber suspension theory," *Journal of Rheology,* vol. 53, pp. 685-705, May-Jun 2009.

[9] K. Ortman, D. Baird, P. Wapperom, and A. Whittington, "Using startup of steady shear flow in a sliding plate rheometer to determine material parameters for the purpose of predicting long fiber orientation," *Journal of Rheology,* vol. 56, pp. 955-981, Jul 2012.

[10] R. S. Bay and C. L. Tucker, "Fiber Orientation in Simple Injection Moldings .1. Theory and Numerical-Methods," *Polymer Composites,* vol. 13, pp. 317-331, Aug 1992.

[11] D. G. Baird, A. P. R. Eberle, G. M. Velez-Garcia, and P. Wapperom, "Fiber orientation kinetics of a concentrated short glass fiber suspension in startup of simple shear flow," *Journal of Non-Newtonian Fluid Mechanics,* vol. 165, pp. 110-119, Feb 2010.

[12] M. Sepehr, P. J. Carreau, M. Moan, and G. Ausias, "Rheological properties of short fiber model suspensions," *Journal of Rheology,* vol. 48, pp. 1023-1048, Sep-Oct 2004.

[13] M. Sepehr, G. Ausias, and P. J. Carreau, "Rheological properties of short fiber filled polypropylene in transient shear flow," *Journal of Non-Newtonian Fluid Mechanics,* vol. 123, pp. 19-32, Oct 15 2004.

[14] M. Sepehr, P. J. Carreau, M. Grmela, G. Ausias, and P. G. Lafleur, "Comparison of rheological properties of fiber suspensions with model predictions," *Journal of Polymer Engineering,* vol. 24, pp. 579-610, Nov-Dec 2004.

[15] R. B. Bird, R. C. Armstrong, and O. Hassager, *Dynamics of polymeric liquids. - 1 : Fluid mechanics*, 2.ed. ed. New York,N.Y.: Wiley, 1987.

[16] C. L. Tucker, J. Wang, and J. F. O'Gara, "An objective model for slow orientation kinetics in concentrated fiber suspensions: Theory and rheological evidence," *Journal of Rheology,* vol. 52, pp. 1179-1200, Sep-Oct 2008.

[17] A. P. R. Eberle, D. G. Baird, P. Wapperom, and G. M. Velez-Garcia, "Obtaining reliable transient rheological data on concentrated short fiber suspensions using a rotational rheometer," *Journal of Rheology,* vol. 53, pp. 1049-1068, Sep-Oct 2009.

[18] H. M. Huynh, "Improved Fiber Orientation Predictions for Injection-Molded Composites," M.S., Mechanical Engineering, University of Illinois at Urbana-Champaign, 1999.

[19] J. Wang, C. A. Silva, J. C. Viana, F. W. J. van Hattum, A. M. Cunha, and C. L. Tucker, "Prediction of fiber orientation in a rotating compressing and expanding mold," *Polymer Engineering and Science,* vol. 48, pp. 1405-1413, Jul 2008.

[20] C. L. Tucker and J. H. Phelps, "An anisotropic rotary diffusion model for fiber orientation in short- and long-fiber thermoplastics," *Journal of Non-Newtonian Fluid Mechanics,* vol. 156, pp. 165-176, Feb 2009.

[21] K. J. Meyer, J. T. Hofmann, and D. G. Baird, "Initial conditions for simulating glass fiber orientation in the filling of center-gated disks," *Composites Part A: Applied Science and Manufacturing,* vol. 49, pp. 192-202, 6// 2013.

[22] S. M. Mazahir, G. M. Vélez-García, P. Wapperom, and D. Baird, "Evolution of fibre orientation in radial direction in a center-gated disk: Experiments and simulation," *Composites Part A: Applied Science and Manufacturing,* vol. 51, pp. 108-117, 8// 2013.

[23] G. M. Velez-Garcia, S. M. Mazahir, P. Wapperom, and D. G. Baird, "Simulation of Injection Molding Using a Model with Delayed Fiber Orientation," *International Polymer Processing,* vol. 26, pp. 331-339, Jul 2011.

[24] D. H. Chung and T. H. Kwon, "Numerical studies of fiber suspensions in an axisymmetric radial diverging flow: the effects of modeling and numerical assumptions," *Journal of Non-Newtonian Fluid Mechanics,* vol. 107, pp. 67-96, Dec 6 2002.

[25] M. Altan, S. Subbiah, S. I. Guceri, and R. B. Pipes, "Numerical Prediction of Three-Dimensional Fiber Orientation in Hele-Shaw Flows," *Polymer Engineering and Science,* vol. 30, pp. 848-859, 1990.

[26] K. H. Han and Y. T. Im, "Numerical simulation of three-dimensional fiber orientation in injection molding including fountain flow effect," *Polymer Composites,* vol. 23, pp. 222-238, Apr 2002.

[27] B. N. Nguyen, S. K. Bapanapalli, J. D. Holbery, M. T. Smith, V. Kunc, B. J. Frame*, et al.*, "Fiber length and orientation in long-fiber injection-molded thermoplastics - Part I: Modeling of microstructure and elastic properties," *Journal of Composite Materials,* vol. 42, pp. 1003-1029, May 2008.

[28] M. Gupta and K. K. Wang, "Fiber Orientation and Mechanical-Properties of Short-Fiber-Reinforced Injection-Molded Composites - Simulated and Experimental Results," *Polymer Composites,* vol. 14, pp. 367-382, Oct 1993.

[29] B. E. VerWeyst and C. L. Tucker, "Fiber suspensions in complex geometries: Flow/orientation coupling," *Canadian Journal of Chemical Engineering,* vol. 80, pp. 1093-1106, Dec 2002.

[30] B. E. VerWeyst, C. L. Tucker, P. H. Foss, and J. F. O'Gara, "Fiber orientation in 3-D injection molded features - Prediction and experiment," *International Polymer Processing,* vol. 14, pp. 409-420, Dec 1999.

[31] S. G. Advani and C. L. Tucker, "The Use of Tensors to Describe and Predict Fiber Orientation in Short Fiber Composites," *Journal of Rheology,* vol. 31, pp. 751-784, Nov 1987.

[32] D. H. Chung and T. H. Kwon, "Fiber orientation in the processing of polymer composites," *Korea-Australia Rheology Journal,* vol. 14, pp. 175-188, Dec 2002.

[33] S. Montgomery-Smith, D. Jack, and D. E. Smith, "The Fast Exact Closure for Jeffery's equation with diffusion," *Journal of Non-Newtonian Fluid Mechanics,* vol. 166, pp. 343-353, Apr 2011.

[34] D. A. Jack, B. Schache, and D. E. Smith, "Neural Network-Based Closure for Modeling Short-Fiber Suspensions," *Polymer Composites,* vol. 31, pp. 1125-1141, Jul 2010.

[35] D. H. Chung and T. H. Kwon, "Improved model of orthotropic closure approximation for flow induced fiber orientation," *Polymer Composites,* vol. 22, pp. 636-649, Oct 2001.

[36] D. H. Chung and T. H. Kwon, "Invariant-based optimal fitting closure approximation for the numerical prediction of flow-induced fiber orientation," *Journal of Rheology,* vol. 46, pp. 169-194, Jan-Feb 2002.

[37] C. W. Hirt and B. D. Nichols, "Volume of Fluid (Vof) Method for the Dynamics of Free Boundaries," *Journal of Computational Physics,* vol. 39, pp. 201-225, 1981.

[38] L. F. R. Shampine, M.W., "The MATLAB ODE Suite," ed. 24 Prime Park Way, Natick, MA 01760: The Mathworks, Inc.

[39] G. M. Velez-Garcia, P. Wapperom, V. Kunc, D. G. Baird, and A. Zink-Sharp, "Sample preparation and image acquisition using optical-reflective microscopy in the measurement of fiber orientation in thermoplastic composites," *Journal of Microscopy,* vol. 248, pp. 23-33, Oct 2012.

[40] G. M. Velez-Garcia, P. Wapperom, D. G. Baird, A. O. Aning, and V. Kunc, "Unambiguous orientation in short fiber composites over small sampling area in a center-gated disk," *Composites Part a-Applied Science and Manufacturing,* vol. 43, pp. 104-113, Jan 2012.

[41] J. T. Hofmann, G. M. Velez-Garcia, D. G. Baird, and A. R. Whittington, "Application and evaluation of the method of ellipses for measuring the orientation of long, semi-flexible fibers," *Polymer Composites,* vol. 34, pp. 390-398, Mar 2013.

4.10 Figures

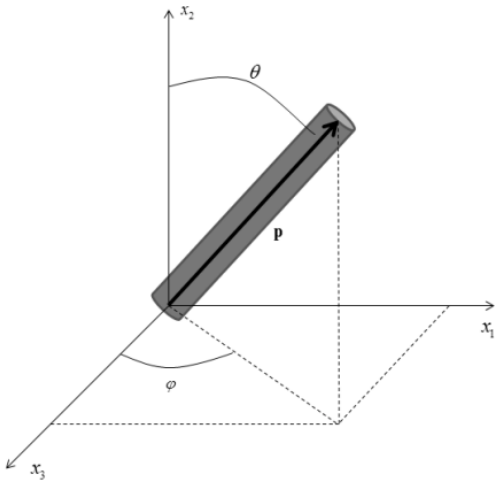

Figure 4.1: Vector definition of rigid-fiber. The fiber is characterized by the vector, $\overline{p}$, as well as the azimuthal and zenith angles, ϕ and θ respectfully.

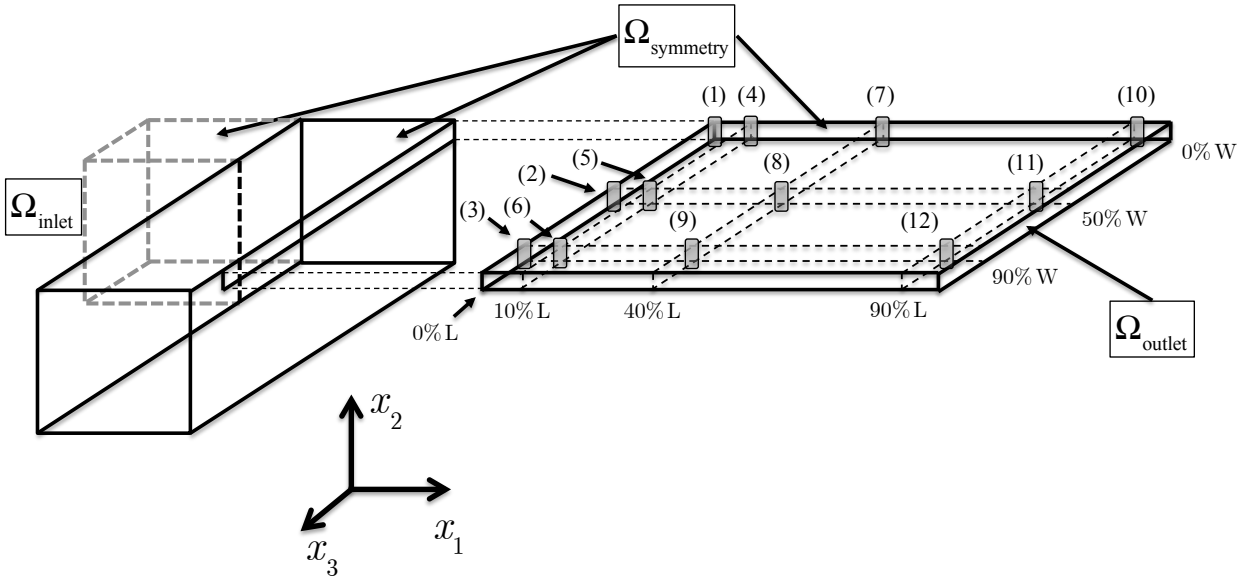

Figure 4.2: End-gated plaque geometry with highlighted regions of interest (1) 0% fill, 0% width, (4) 10% fill, 0% width, (7) 40% fill, 0% width, (10) 90% fill, 0% width, (2) 0% fill, 50% width, (5) 10% fill, 50% width, (8) 40% fill, 50% width, (11) 90% fill, 50% width, (3) 0% fill, 90% width, (6) 10% fill, 90% width, (9) 40% fill, 90% width, (12) 90% fill, 90% width.

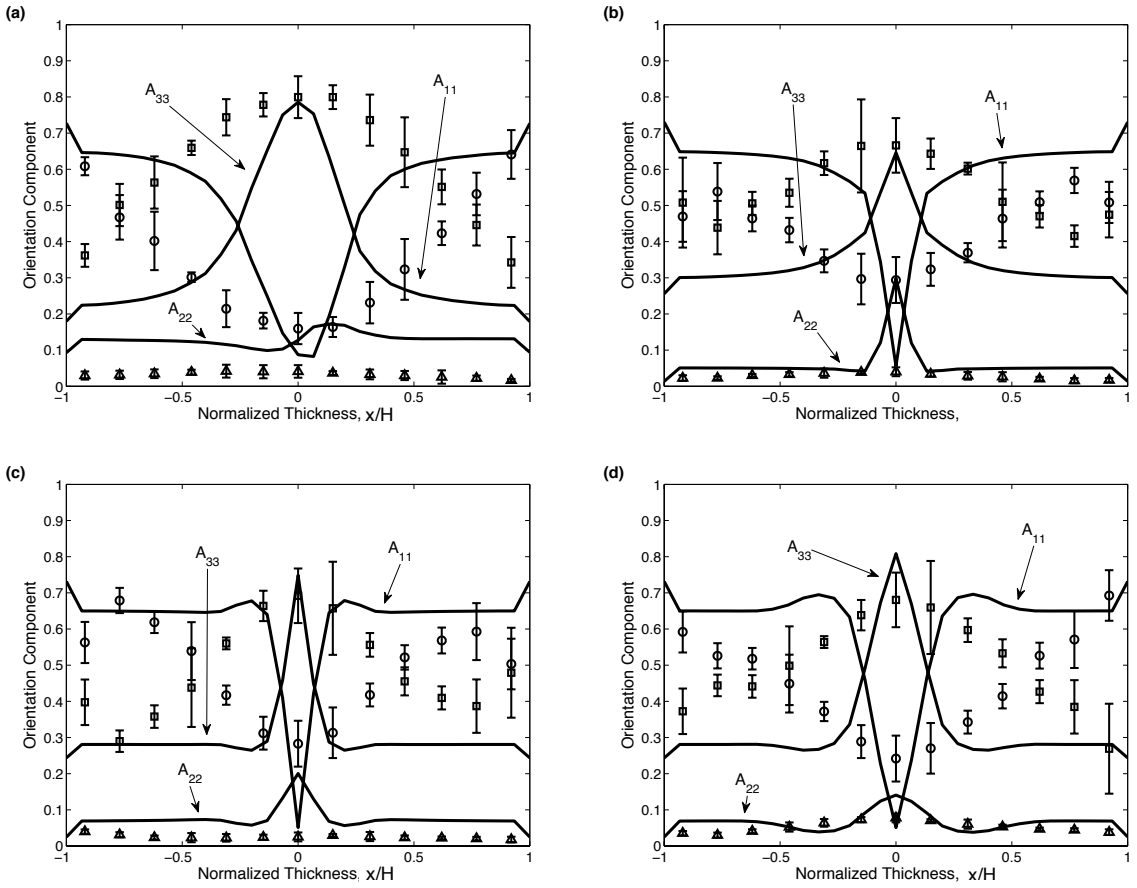

Figure 4.3: Model predictions of the SRF model using orientation parameters obtained from fitting stress growth at the startup of shear flow. Predictions are compared to experimentally determined fiber orientation (A_{11} - O, A_{22} - Δ, A_{33} - □) at 0% of the mold width (centerline) at (a) 0% mold fill (position (4) in Figure 4.2), (b) 10% mold fill (position (7) in Figure 4.2), (c) 40% mold fill (position (10) in Figure 4.2) and (d) 90% mold fill (position (13) in Figure 4.2). 2H = 1.55 mm.

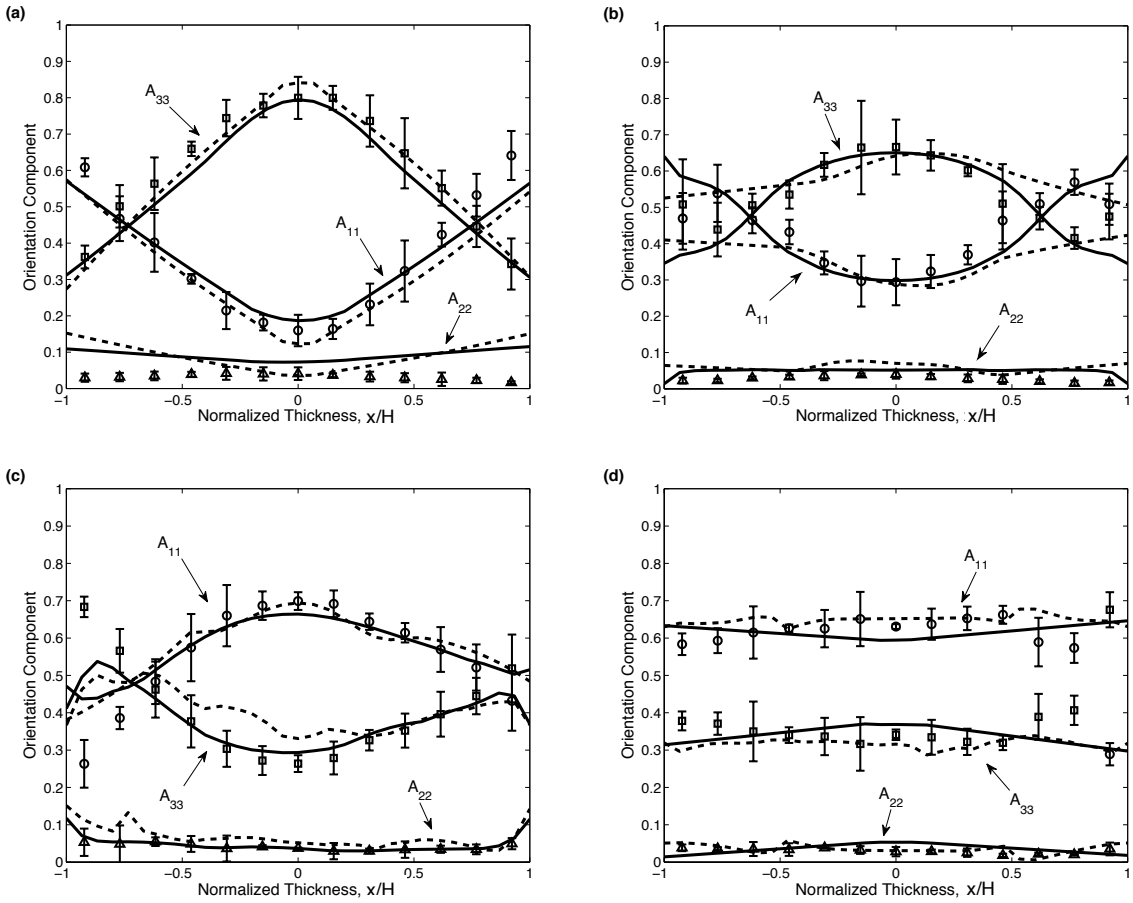

Figure 4.4: Model predictions of SRF (solid) and RSC (dashed) models compared to experimentally determined fiber orientation (A_{11} - ¢, A_{22} - Δ, A_{33} - □) at 0% of the mold width (centerline) at (a) 0% mold fill along the centerline (position (4) in Figure 4.2), (b) 10% mold fill along the centerline (position (7) in Figure 4.2), (c) 0% mold fill at 50% width (position (5) in Figure 4.2) and (d) 40% mold fill at 50% width (position (11) in Figure 4.2). 2H = 1.55 mm.

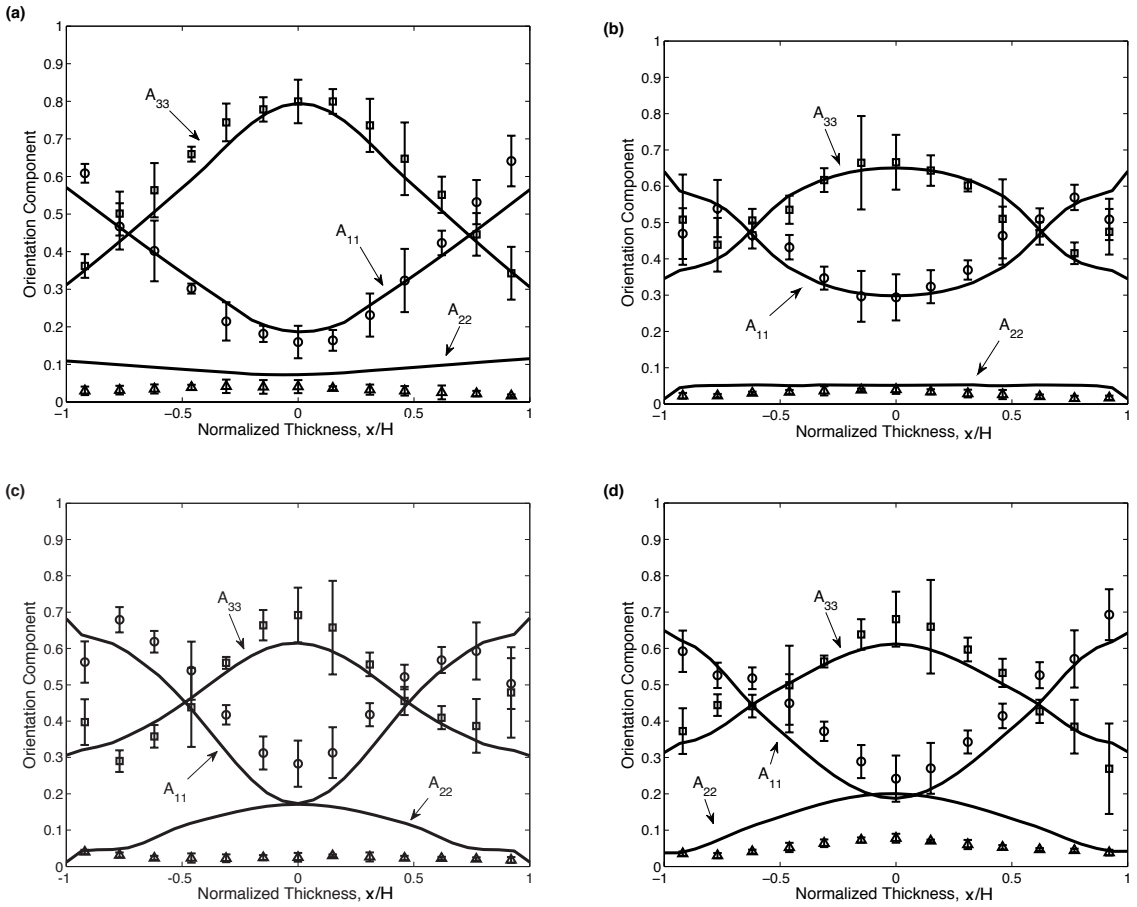

Figure 4.5: Model predictions of SRF (solid) model compared to experimentally determined fiber orientation (A_{11} - ¢, A_{22} - Δ, A_{33} - □) at 0% of the mold width (centerline) at (a) 0% mold fill (position (4) in Figure 4.2), (b) 10% mold fill (position (7) in Figure 4.2), (c) 40% mold fill (position (10) in Figure 4.2) and (d) 90% mold fill (position (13) in Figure 4.2). 2H = 1.55 mm.

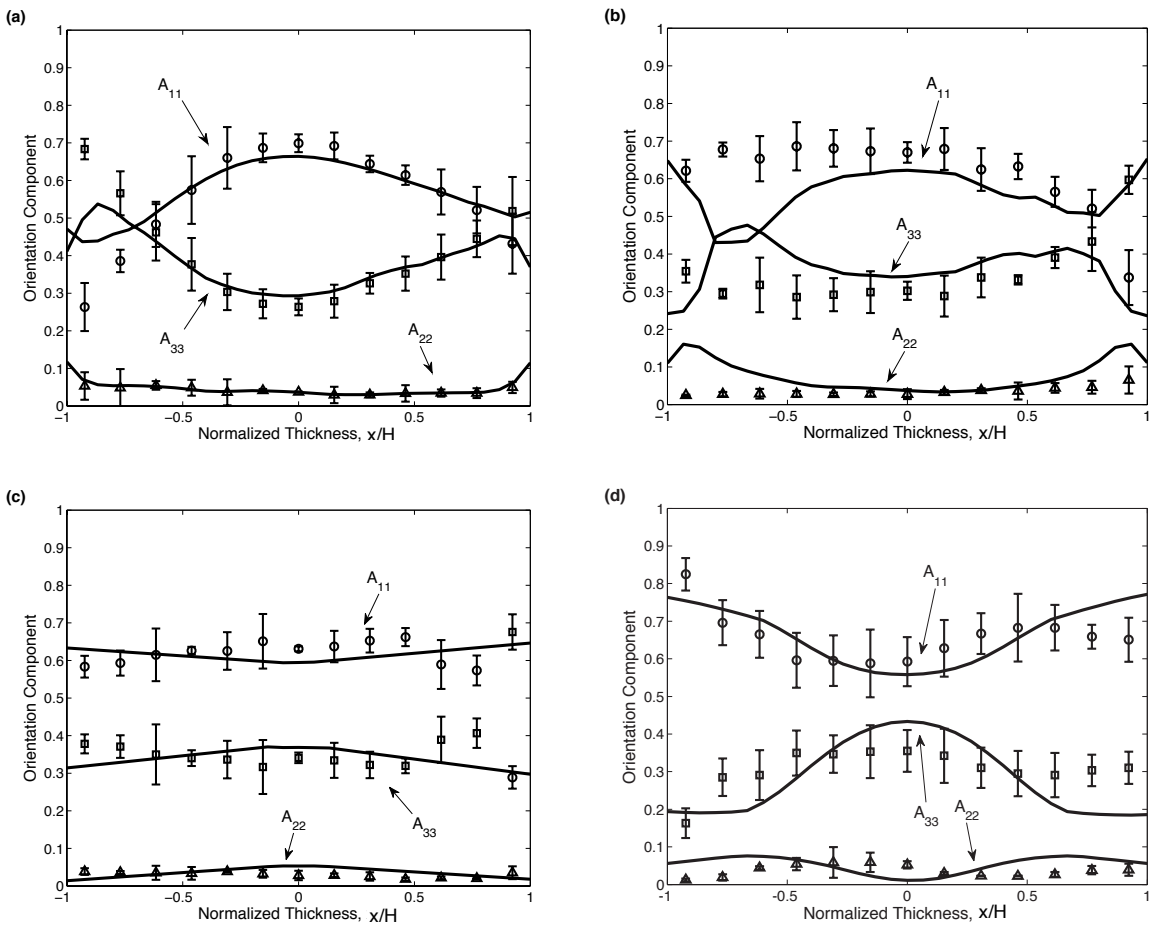

Figure 4.6: Model predictions of SRF (solid) model compared to experimentally determined fiber orientation (A_{11} - ¢, A_{22} - Δ, A_{33} - □) at 50% of the mold width (centerline) at (a) 0% mold fill (position (5) in Figure 4.2), (b) 10% mold fill (position (8) in Figure 4.2), (c) 40% mold fill (position (11) in Figure 4.2) and (d) 90% mold fill (position (14) in Figure 4.2). 2H = 1.55 mm.

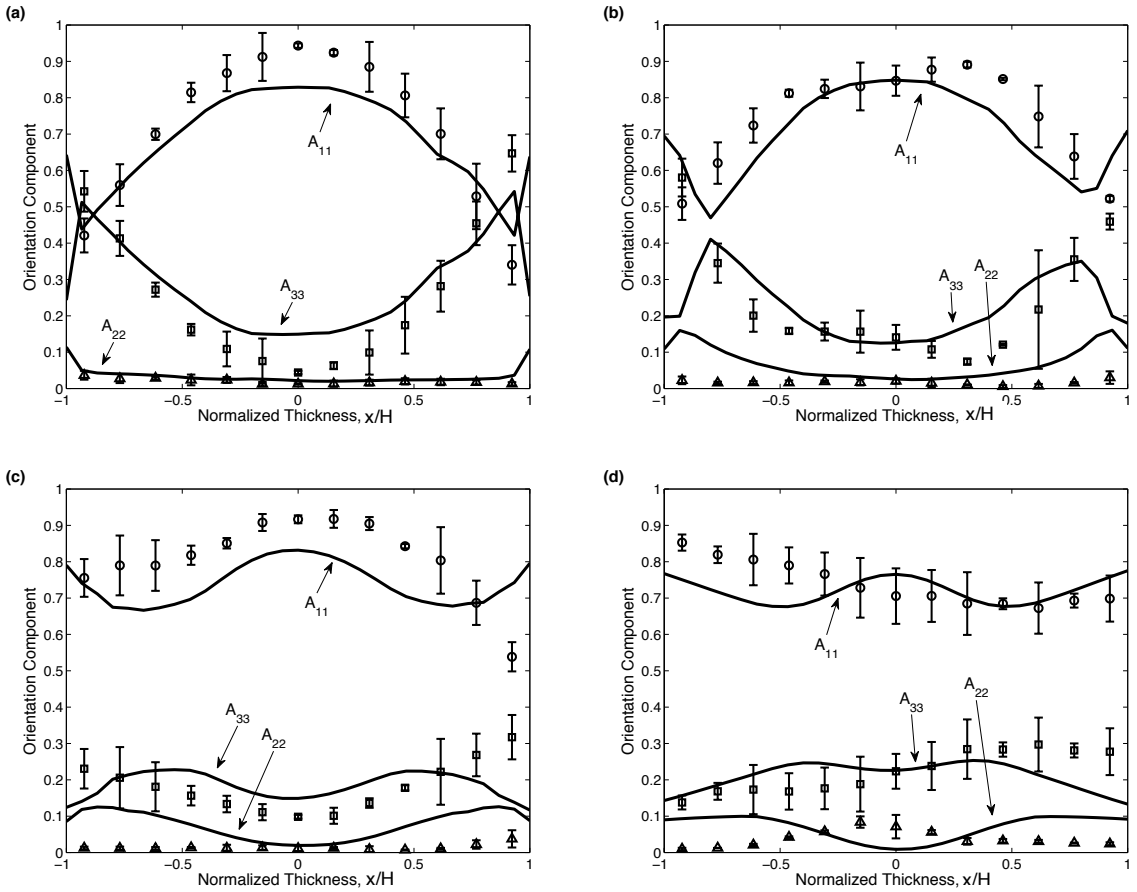

Figure 4.7: Model predictions of SRF (solid) model compared to experimentally determined fiber orientation (A_{11} - ¢, A_{22} - Δ, A_{33} - □) at 50% of the mold width (centerline) at (a) 0% mold fill (position (6) in Figure 4.2), (b) 10% mold fill (position (9) in Figure 4.2), (c) 40% mold fill (position (12) in Figure 4.2) and (d) 90% mold fill (position (15) in Figure 4.2). 2H = 1.55 mm.

Parameter	Value
$\rho\left(\mathrm{kg/m^3}\right)$	1100
$\eta\left(\mathrm{Pa\cdot s}\right)$	4814
$\lambda\left(\mathrm{s^{-1}}\right)$	0.2777
n	0.9090
$C_P\left(\mathrm{J/kg\cdot K}\right)$	2800
$k\left(\mathrm{W/m\cdot K}\right)$	0.234
$\alpha\left(\mathrm{K}\right)$	4220
$T_\alpha\left(\mathrm{K}\right)$	427.23
$T_{\mathrm{Wall}}\left(\mathrm{K}\right)$	363
$T_{\mathrm{Inlet}}\left(\mathrm{K}\right)$	463

Table 4.1: Material properties for non-isothermal finite element simulation.

Parameter	SRF	RSC
κ	0.0297	0.0910
C_I	0.0481	0.0658

Table 4.2: Fiber orientation simulation parameters.

Chapter 5

Prediction of Orientation of Long Semi-Flexible Glass Fiber Orientation during the Injection Molding of an End-Gated Plaque

Preface

This chapter describes the prediction of long glass fiber orientation in the filling of a three-dimensional end-gated plaque using a rigid and semi-flexible fiber orientation model and comparing to experimentally determined values of orientation. This chapter is organized as a manuscript for publication in Composites Part A: Applied Science and Manufacturing.

5 Prediction of Orientation of Long Semi-Flexible Glass Fiber Orientation during the Injection Molding of an End-Gated Plaque

Kevin J. Meyer[1], John T. Hofmann[2], Donald G. Baird[1]* and Vlastimil Kunc[3]

1. Department of Chemical Engineering, Virginia Tech, Blacksburg, VA 24061
2. Macromolecule and Interfaces Institute, Virginia Tech, Blacksburg, VA 24061
3. Oak Ridge National Laboratory, P.O. Box 2009, Oak Ridge, TN 37831

* Corresponding Author: Tel. +1 540 231 5998; Fax +1 540 231 2732.
Email Address: dbaird@vt.edu (D.G. Baird)
Present Address:
Department of Chemical Engineering
133 Randolph Hall
Virginia Tech
Blacksburg, VA 24061

5.1 Abstract

This work is concerned with predicting the orientation of long $(L > 1\text{ mm})$ glass fibers (LGF) during injection molding in an end-gated plaque (EGP), which represents a basic 3-dimensional geometry. Previous EGP simulations have provided orientation predictions results only within the mold cavity along the centerline of the mold. This paper reports on a method to simulate the entire domain of the EGP to obtain fiber orientation predictions without making assumptions about fiber orientation entering the mold and comparing results to experimental data both along and away from the centerline. Furthermore, rigid and semi-flexible fiber orientation models are employed to predict LGF orientation. Model parameters are obtained through both a rheological and experimental fitting procedure. The semi-flexible fiber model is observed to be in better quantitative agreement than the rigid fiber model at predicting observed experimental fiber orientation at a number of positions in the EGP.

5.2 Introduction

A "long glass fiber" is defined here for the duration of this work as a fiber that exhibits the ability to deform during processing. This bending can effect a fiber's ability to orient in a given flow field and alter the material's microstructure and thus altering the bulk material properties. The dimensionless parameter suggested by Swtizer and Klingenberg [1] is adopted in this work to quantify the potential for bending and is given by Eq. (5.1):

$$F^{\text{eff}} = \frac{64\eta_m \dot{\gamma} a_r^4}{E_Y \pi} \tag{5.1}$$

In Eq. (5.1) η_m is the matrix viscosity, $\dot{\gamma}$ is the magnitude of the second invariant of the rate of deformation, a_r is the aspect ratio of the particle and E_Y is the Young's modulus. In literature a

glass fiber is typically considered long when it's length is above 1 mm [2]. Calculating an effective fiber flexibility using this convention and the properties of an e-glass fiber suggests that a rigid glass fiber exists when $F^{\text{eff}} < 51$ and a flexible glass fiber exists when $F^{\text{eff}} > 51$. Therefore, when the effective fiber flexibility parameter exceeds a value of 51, flexibility may play a role in dictating a material's microstructure.

Predicting the orientation of a concentrated rigid glass fiber suspension typically starts with a modified form of Jeffery's equation for the motion of an ellipsoidal particle in a viscous medium [3]. Folgar and Tucker [4] modified Jeffery's equation by adding a term based on isotropic diffusivity that was proportional to the velocity gradient to account for fiber-fiber interaction in a concentrated fiber system. The isotropic rotary diffusion term does depend on a fiber interaction coefficient, C_I, that has been studied through both theoretical and experimental means [5, 6]. The Folgar-Tucker (FT) model has shown good qualitative agreement in the Hele-Shaw region with experimental data and is thus a popular choice simulating short glass fiber orientation [6, 7].

Stress growth experiments in the startup of simple shear flow have shown that the orientation of rigid fibers in concentrated suspension evolves more slowly than predicted by the Folgar-Tucker model [8, 9]. To more accurately reflect the observed transient fiber orientation a "slip" parameter, κ, was suggested by Sepehr et al. [10, 11] and Eberle et al. [8]. The strain reduction factor (SRF) model more closely agreed with experimental data but is not objective [12]. Wang et al. [13] developed the reduced strain closure (RSC) model as an objective form of the SRF model where the closure approximation is modified eliminating the objectivity problem while including the slower orientation kinetics observed in experimental values of orientation. Phelps and Tucker [14] have developed a form of the RSC model (ARD-RSC) which accounts for the anisotropy in fiber interactions but requires additional fitting efforts in determining six model coefficients. Even though the SRF model is non-objective and assumes isotropy, it has been shown to be useful in describing the evolution of orientation in simple flows and in more general flows [14-16]. Furthermore, Mazahir et al. [17] showed that for Hele-Shaw flows and Meyer et al. [18] showed for flows including the gate region and advancing front that the SRF and RSC model provide similar predictions for short glass fibers in the center-gated disk and end-gated plaque.

During the processing of concentrated long glass fiber (LGF) suspensions further modification to the orientation equations may be necessary. Strautins and Latz [19] proposed a two rod continuum based approach for modeling deformable fibers assuming that suspension was dilute ($\phi < 1/a_r^2$). This two-rod model incorporated the effect that fiber length and fiber flexibility play in predicting LGF orientation. Ortman et al. [15] adapted the dilute theory to a concentrated suspension by adding the isotropic rotary diffusion term from the Folgar-Tucker model and saw good agreement with predicted fiber orientation in simple shear flow [4]. Ortman et al. [16] observed an increased agreement between predicted LGF orientation values and experimentally observed LGF orientation values up to 40% of the mold fill but used experimental orientation values as initial conditions to the orientation predictions. Meyer et al. [20] used a semi-flexible fiber model and incorporated the gate region and advancing front and observed that this semi-flexible fiber model was more successful at predicting experimental LGF orientation in a CGD geometry at a number of positions.

The prediction fiber orientation in commercially relevant complex test geometries has been the subject of little work for LGF systems. Meyer et al. [20] investigated LGF orientation in a CGD and saw an increased agreement with experimental orientation data when using an orientation model that included terms taking into account a fiber's ability to mildly deform and including the gate region and advancing front. Nguyen et al. [21] simulated long glass fiber orientation with a rigid fiber model in an EGP and also found reasonable agreement with experimental orientation data but only made comparisons along the centerline of the mold. Furthermore, the work of Nguyen et al. [21] in the EGP geometry had an average fiber length of $L_W \approx 1.83\text{mm}$ with a strong majority of the population of fibers being under 1 mm in length which may have contributed to a rigid fiber model developed for short fiber systems accurately predicting long fiber orientation. Therefore, there is a void in current literature for a comprehensive analysis of the ability of current predictive models to predict fiber orientation both along and away from the centerline of the EGP geometry.

The purpose of this work is to predict long semi-flexible fiber orientation $\left(L_W \approx 3.90\text{mm}\right)$ in a commercially relevant end-gated plaque geometry using both a rigid and semi-flexible fiber orientation model and compare the predicted results with experimentally obtained values of fiber orientation both along and away from the centerline and drawing conclusions as to the accuracy of the predictions. The gate and mold of the EGP are simulated as one continuous domain so that no assumptions about fiber orientation or inlet velocity profile have to be made at the mold entrance which has been shown to affect predicted orientation values [20]. The solution for velocity fields and fiber orientation are decoupled so that a two-step process to calculate orientation is invoked [20, 22]. The finite element method in the ANSYS Polyflow© environment is used for the non-isothermal velocity field solution including the advancing front. The solution of the rigid and semi-flexible fiber orientation equations uses finite difference methods written in MATLAB and C. Fiber orientation predictions are carried out using orientation parameters $\left(\kappa \text{ and } C_I\right)$ obtained through fitting transient shear stress over shoots from the startup of shear flow experiments and through fitting experimental orientation data from injection molding experiments. . Rigid and semi-flexible fiber orientation model predictions are compared to experimental data obtained through the method of ellipses given by Vélez-García [23] at a number of positions in the EGP mold to access the predictive ability of the modeling techniques.

5.3 Governing Equations

5.3.1 Equations of Motion and Energy

The flow inside the cavity is assumed incompressible and laminar (calculations of Reynolds numbers in the cavity produced values of $\text{Re} \le 10^{-3}$). From these assumptions the continuity equations and the equation of motion are written as Eq. (5.2) and Eq. (5.3) where $\underline{v}$ is the velocity vector, P is the isotropic pressure and $\underline{\underline{\tau}}$ is the extra stress tensor:

$$\nabla \cdot \underline{v} = 0 \tag{5.2}$$

$$-\nabla P + \nabla \cdot \underline{\underline{\tau}} = 0 \tag{5.3}$$

Eq. (5.3) requires the choice of an extra stress tensor representation and is discussed in a following section.

The energy equation is solved to include temperature effects in the form given in Eq. (5.4) where ρ is the fluid density, C_P is the specific heat capacity, β is the thermal conductivity, $\underline{\underline{\tau}}$ is the extra stress tensor, $\underline{\underline{\dot{\gamma}}}$ is the rate of deformation tensor and D/Dt is the material derivative $\left(D/Dt = \partial/\partial t + \underline{v}\cdot\nabla\right)$:

$$\rho C_P \frac{DT}{Dt} = \beta \nabla^2 T + \underline{\underline{\tau}} : \underline{\underline{\dot{\gamma}}} \tag{5.4}$$

The viscosity of the fluid was assumed to be temperature dependent through the use of an Arrhenius-type relationship given in Eq. (5.5) where $\eta(\dot{\gamma})$ is the shear dependent viscosity, α is the ratio of activation energy to Boltzmann's constant $\left(\alpha = E_a/k\right)$ and T_α is a reference temperature:

$$\eta(\dot{\gamma},T) = \eta(\dot{\gamma})\exp\left[\alpha\left(\frac{1}{T}-\frac{1}{T_\alpha}\right)\right] \tag{5.5}$$

The values for the temperature dependence of the system can be found in Table 5.1.

5.3.2 Stress Tensor Representation

During the fitting of constants for long glass fiber systems, the Dinh-Armstrong stress tensor model, multiple solutions were obtained suggesting that the stress tensor may not accurately capture all of the phenomena occurring in the concentrated fiber system [24]. This suggests that the rheology of the concentrated suspension is more complex than current models can predict and thus a decoupled approach for the stress and orientation tensors is used in this work. From the perspective of the fibers, the major drag experienced is from the polymer melt. The effect of fibers in the suspension was included by incorporating the fiber loading and fiber-fiber interaction contributions into the simulations through the values of κ and C_I so that only the neat matrix properties are used to predict the velocity fields.

$$\underline{\underline{\tau}} = \eta(\dot{\gamma})\underline{\underline{\dot{\gamma}}} \tag{5.6}$$

A Generalized Newtonian Fluid was chosen as the constitutive relation to represent the matrix used in this study given by the relation in Eq. (5.6) where $\underline{\underline{\tau}}$ is the extra stress tensor, $\eta(\dot{\gamma})$ is an empirical relation for describing the viscosity as a function of shear rate and $\underline{\underline{\dot{\gamma}}}$ is the rate of strain tensor. The stress tensor is taken as positive for tensile stresses following the mechanics sign convention.

$$\eta(\dot{\gamma}) = \eta_0\left[1+\left(\lambda\dot{\gamma}\right)^a\right]^{\frac{n-1}{a}} \tag{5.7}$$

The Carreau-Yasuda model was used to account for the shear-thinning nature of neat matrix where η_0 is the zero shear viscosity, λ is the parameter which governs the onset of shear thinning, n is a parameter which governs the degree of shear thinning within the matrix and a is a parameter that describes the transition region between the zero-shear region and the power-law

region and is given by Eq. (4.6). The values used in the Carreau-Yasuda model are given in Table 5.1 and were estimated using a rotation rheometer (RMS-800, Rheometrics, Inc.).

5.3.3 Rigid Fiber Orientation Equation

The orientation tensor of Advani and Tucker [25] is a compact way of representing fiber orientation and is defined by Eq. (5.8) where $\underline{\underline{A}}$ is the second-order orientation tensor, $\underline{p}$ is a vector drawn through the longitudinal axis of the fiber (shown in Figure 5.1) and $\psi\left(\underline{p},t\right)$ is the probability distribution function and t is time:

$$\underline{\underline{A}} = \iint \underline{pp}\,\psi\left(\underline{p},t\right)d\underline{p} \tag{5.8}$$

Within in the equations of orientation a fourth order tensor also arises shown in Eq. (5.9):

$$\underline{\underline{A}}_4 = \iint \underline{pppp}\,\psi\left(\underline{p},t\right)d\underline{p} \tag{5.9}$$

The fourth-order tensor requires the use of a closure approximation, for which there are a number of choices summarized in Chung and Kwon [26]. For the duration of this work the invariant-based optimal fitting (IBOF) closure is used because has been shown by Chung and Kwon to produce results very similar to directly calculating the orientation distribution function and is more computationally efficient [27].

In order to efficiently predict fiber orientation in molding geometries, Jeffery's equation has been used with the addition of an interaction term to account for fiber-fiber interaction known as the modified Folgar-Tucker model [3, 4]. Sepehr et al. [10] proposed a modified form of the Folgar-Tucker model shown in Eq. (5.10) where $\dot{\underline{\underline{A}}}$ is the material derivative $\left(\partial\underline{\underline{A}}/\partial t + \underline{v}\cdot\nabla\underline{\underline{A}}\right)$ of the 2nd order orientation tensor $\underline{\underline{A}}$, $\underline{\underline{W}}$ is the vorticity $\left(\underline{\underline{W}} = \tfrac{1}{2}\left[\nabla\underline{v}^T - \nabla\underline{v}\right]\right)$, $\underline{\underline{D}}$ is the rate of deformation $\left(\underline{\underline{D}} = \tfrac{1}{2}\left[\nabla\underline{v}^T + \nabla\underline{v}\right]\right)$, $\underline{\underline{A}}_4$ is the 4th order orientation tensor, $\dot{\gamma}$ is the magnitude of the rate of deformation tensor $\left(\dot{\gamma} = \sqrt{2\left(\underline{\underline{D}}:\underline{\underline{D}}\right)}\right)$, and $\underline{\underline{I}}$ is the identity tensor and $\nabla\underline{v} = \partial v_j/\partial x_i$:

$$\dot{\underline{\underline{A}}} = \kappa\left[\underline{\underline{W}}\cdot\underline{\underline{A}} - \underline{\underline{A}}\cdot\underline{\underline{W}} + \xi\left(\underline{\underline{D}}\cdot\underline{\underline{A}} + \underline{\underline{A}}\cdot\underline{\underline{D}} - 2\underline{\underline{A}}_4 : \underline{\underline{D}}\right) + 2C_I\dot{\gamma}\left(\underline{\underline{I}} - 3\underline{\underline{A}}\right)\right] \tag{5.10}$$

The variables κ and C_I are empirical parameters used to represent fiber-fiber interaction and fiber loading and are commonly fit to experimental data [28]. The value of C_I will dictate how highly fibers align in the flow direction (lower value of C_I will yield higher flow aligned fibers) while the value of κ accounts for the overall slower orientation evolution seen in concentrated fiber systems and has a value of between zero and unity.

5.3.4 Semi-Flexible Fiber Orientation Equations

Strautins and Latz [19] developed orientation equations for a dilute semi-flexible fiber suspension using orientation tensors shown in Eq. (5.11) to Eq. (5.13) where $\underline{p}$ and $\underline{q}$ represent the two vectors of the bent fiber (shown in Figure 5.2) and $\psi\left(\underline{p},\underline{q},t\right)$ is the probability distribution function for an individual rod:

$$\underline{\underline{A}} = \iint \underline{p}\underline{p}\,\psi\left(\underline{p},\underline{q},t\right)d\underline{p}\,d\underline{q} \tag{5.11}$$

$$\underline{\underline{B}} = \iint \underline{p}\underline{q}\,\psi\left(\underline{p},\underline{q},t\right)d\underline{p}\,d\underline{q} \tag{5.12}$$

$$\underline{C} = \iint \underline{p}\,\psi\left(\underline{p},\underline{q},t\right)d\underline{p}\,d\underline{q} \tag{5.13}$$

A final tensor of interest is the end-to-end orientation tensor, $\underline{\underline{R}}$, that represents the average orientation of a slightly deformed fiber and is given by Eq. (5.14):

$$\underline{\underline{R}} = \frac{\left(\underline{\underline{A}} - \underline{\underline{B}}\right)}{1 - \mathrm{tr}\left(\underline{\underline{B}}\right)} \tag{5.14}$$

Ortman et al. [15] adapted the orientation equations given by Strautins and Latz [19] to concentrated suspensions by including the isotropic rotary diffusion term and slip parameter from previous work [6, 10]. The adaptation utilizes the orientation tensors in Eqs. (5.11) - (5.13) and is given by Eqs. (5.15) - (5.18) where l_b is the length of one rod of the two rod system:

$$\begin{aligned}\dot{\underline{\underline{A}}} = \kappa\Big[&\underline{\underline{W}}\cdot\underline{\underline{A}} - \underline{\underline{A}}\cdot\underline{\underline{W}} + \xi\left(\underline{\underline{D}}\cdot\underline{\underline{A}} + \underline{\underline{A}}\cdot\underline{\underline{D}} - 2\underline{\underline{A}}_4 : \underline{\underline{D}}\right) - 6C_I\dot{\gamma}\left(\underline{\underline{A}} - \tfrac{1}{3}\underline{\underline{I}}\right) + \ldots \\ &\frac{l_b}{2}\left[\underline{C}\underline{m} + \underline{m}\underline{C} - 2\left(\underline{m}\cdot\underline{C}\right)\underline{\underline{A}}\right] + 2k\left(\underline{\underline{B}} - \underline{\underline{A}}tr\left(\underline{\underline{B}}\right)\right)\Big]\end{aligned} \tag{5.15}$$

$$\begin{aligned}\dot{\underline{\underline{B}}} = \kappa\Big[&\underline{\underline{W}}\cdot\underline{\underline{B}} - \underline{\underline{B}}\cdot\underline{\underline{W}} + \xi\left(\underline{\underline{D}}\cdot\underline{\underline{B}} + \underline{\underline{B}}\cdot\underline{\underline{D}} - 2\left(\underline{\underline{D}} : \underline{\underline{A}}\right)\underline{\underline{B}}\right) - 4C_I\dot{\gamma}\underline{\underline{B}} + \ldots. \\ &\frac{l_b}{2}\left[\underline{C}\underline{m} + \underline{m}\underline{C} - 2\left(\underline{m}\cdot\underline{C}\right)\underline{\underline{A}}\right] + 2k\left[\underline{\underline{A}} - \underline{\underline{B}}\ \mathrm{tr}\left(\underline{\underline{B}}\right)\right]\Big]\end{aligned} \tag{5.16}$$

$$\dot{\underline{C}} = \kappa\left[\nabla\underline{v}^T\cdot\underline{C} - \left(\underline{\underline{A}} : \nabla\underline{v}^T\right)\underline{C} + \frac{l_b}{2}\left[\underline{m} - \underline{C}\left(\underline{m}\cdot\underline{C}\right)\right] - k\underline{C}\left[1 - \mathrm{tr}\left(\underline{\underline{B}}\right)\right] - 2C_I\dot{\gamma}\underline{C}\right] \tag{5.17}$$

$$\underline{m} = \sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3}\frac{\partial^2 v_i}{\partial x_j\,\partial x_k}A_{jk}\underline{\delta}_i \tag{5.18}$$

The vector $\bar{m}$ accounts for the bending of a fiber due to the flow field surrounding that fiber if the second derivative of the velocity gradient is present.

In the semi-flexible fiber model, the fiber flexibility parameter can modify the evolution of fiber orientation when a flow field is applied. The fiber flexibility parameter, k, is a coefficient that attempts to account for the physical bending that may occur in the system of fibers. For this work the fiber flexibility parameter is calculated through a relation derived from a beam simply supported on both ends with a point force applied at the center. The value of k is calculated at each node on the mesh through the use of Eq. (5.19):

$$k = \frac{E_Y}{64\eta_m a_r^3} \tag{5.19}$$

In Eq. (5.19) k is the fiber flexibility at a particular mesh node, E_Y is the Young's modulus of the fiber, η_m is the viscosity of the matrix at a particular mesh node (given by Eq. (5.7)) and a_r is the aspect ratio of the fiber. As the value of k increases, the semi-flexible model behaves more like the rigid fiber model and in the limit of $k \to \infty$ the semi-flexible model parallels the rigid fiber model. Conversely, as the value of k decreases the fiber becomes much more flexible and in the limit of $k \to 0$ the fiber is completely flexible. In the equations presented above, the flexibility of the fiber is initiated in the hydrodynamic contributions given by Eq. (5.18) and is due to the bending that may occur due to gradients in the flow field.

The semi-flexible fiber model also includes a length term that may further modify the evolution of orientation when a flow field is applied. The length of one half of the semi-flexible rod, l_b, is used in all three of the equations for the semi-flexible fiber model (Eqs. (5.15) -(5.17)) and incorporates the length of the fiber system into the orientation equations which has not been done in previous modeling rigid modeling techniques.

5.4 Numerical Method

5.4.1 Fiber Orientation Prediction Method

The filling of the cavity was simulated using the ANSYS finite element (FEM) software package. The entrance to the gate region was meshed using 31 (thickness) x 30 (length) x 20 (width) elements. The gate region was meshed using 31 (thickness) x 31 (length) x 50 (width) hexagonal elements. The mold was meshed using 31 (thickness) x 300 (length) x 50 (width) hexagonal elements. Increasing the number of elements did not change the results on the scale of the plots.

The time-stepping scheme chosen to solve for the transient mold filling is the volume of fluid method inside the ANSYS Polyflow environment that solves a transport equation to capture the air-polymer interface with an internally managed variable time step. The flow and transport equations are solved in a decoupled fashion at each time step. The solution of the transport equation uses streamline upwinding and interpolates between elements with a linear sub-element interpolation method. The solution of the non-isothermal system required the use of an evolution scheme at each time step inside the ANSYS Polyflow environment to control the viscous dissipation and convection term associated with Eq. (5.4).

The fiber orientation equations were solved using 1st order accurate finite differences (FDM) in MATLAB (The Mathworks Inc., ver. 7.4) and C. By discretizing in space the convected part of the derivative, the equations of orientation may be rewritten as a function of time only. This reduces the set of partial differential equations to a system of non-linear ordinary differential equations. The ordinary differential equations are solved using a variable step size backwards differences implementation of the Kopfenstein-Shampine family [29].

5.4.2 Boundary Conditions

The non-isothermal FEM simulations of the EGP (shown in Figure 5.3) require specified boundary conditions for both the flow and heat equations. The inlet boundary requires an inlet

fluid temperature $\left(T_{in} = 463\text{K}\right)$ and volumetric flow rate $\left(Q_{in} = 3225\,\frac{\text{mm}^3}{\text{s}}\right)$ and the velocity field is assumed to be fully developed. The symmetry condition is specified on the boundary Ω_{sym} in Figure 5.3. At the walls of the mold, specified by Ω_{wall} in Figure 5.3, both the mold wall temperature and the no slip condition $\left(T_{wall} = 363\,\text{K}, \underline{v} = \underline{0}\right)$ were specified. Furthermore, a zero traction vector is defined at the fluid front i.e. $\underline{n} \bullet \underline{\underline{\sigma}} \bullet \underline{n} = 0$ and $\underline{\underline{\sigma}} \bullet \underline{n} - \left(\underline{n} \bullet \underline{\underline{\sigma}} \bullet \underline{n}\right)\underline{n} = \underline{0}$. These free boundary conditions are prescribed where the fluid fraction is zero. The temperature equation at the fluid front is solved using a prescribed temperature of 300K.The FDM simulation required initial values for the orientation tensor to be specified at the inlet to the sprue. The orientation of the fibers entering the inlet boundary, Ω_{inlet}, was prescribed as completely random $\left(\underline{\underline{A}} = \frac{1}{3}\underline{\underline{I}}\right)$ but this choice did not influence orientation predictions in the mold.

5.5 Experimental Conditions

5.5.1 Geometry and Processing Methods

End-gated plaques (EGP) were formed for the experimental evaluation of fiber orientation using 30 weight percent LGF in a polypropylene matrix (LNP Verton MV006S) provided by Sabic Innovation Plastics. The EGP has a sprue length of 65mm with an initial radius of 1.45mm and a final radius at the gate of 1.75mm. The gate region of the plaque has dimensions of 80.68mm (width) by 6.25mm (height) by 6.33mm (length) and the mold region of the plaque has dimensions of 75.05mm (width) by 1.55mm (height) by 77.65mm (length).

An injection molding machine was utilized to mold a series of EGP's. The temperatures of the feed, compression and metering zones within the injection molding machine (Arburg Allrounder, Model 221-55-250) were set to 190 °C, 210 °C, and 210 °C, respectively, while the mold temperature was held constant at 79 °C. The plaques were molded using a fill time of 2 seconds (fast fill times were chosen to minimize non-isothermal effects) with a backpressure of approximately 20 MPa. Additionally, the end-gated mold was filled using a 90% short shot in order to mitigate any potential packing effects, and the resulting plaques had an average final length of 68.65 ± 1.87 mm. Furthermore, all plaques were left to cool in the closed mold for a period of twenty minutes prior to removal in order to reduce warping.

5.5.2 Fiber Length Distribution Determination

The fiber length distribution (FLD) was determined to verify that the average lengths of the fibers in the EGP were indeed in the long fiber regime that is typically defined as $l > 1\text{mm}$. Methods described by Nguyen et al. [30] were used to determine the fiber length distribution of approximately 2000 fibers. The method involves taking a representative sample of the suspension of the injection-molded sample and burning off the polymer matrix in a high temperature furnace leaving only the glass fiber mat behind. A small amount of epoxy resin was applied to the glass fiber matrix to secure a population of fibers and any fibers not secured to the epoxy were carefully removed. The remaining fiber/epoxy sample was returned to the furnace to remove the epoxy and leave only the glass fibers. Finally, the remaining fibers are analyzed through optical techniques in order to determine the experimental FLD. The post-processing number average fiber length was $3.90\text{mm} \pm 0.11\text{mm}$ and the average fiber diameter was

determined to be $14.2\mu m$. This wider distribution of LGF is important because of the relevance to fiber length distributions found in larger commercially relevant injection molded parts [31].

5.5.3 Fiber Orientation Measurements

A set of five representative plaques were selected and prepared for analysis at multiple locations according to procedure described in detail by Velez-Garcia et al. [32]. The chosen inspection points to obtain experimental values of orientation were 0%, 10%, 40%, and 90% of length of the mold, and along the centerline, 50% of the half-width of the mold, and 90% of the half-width of the mold. After sample preparation via polishing, an optical microscope with a motorized stage and image-stitching software (Nikon Eclipse LV100, NIS-Elemets Basic Research software, v.3.10) was subsequently utilized to image each inspection point at 20X magnification.

The experimental components of the orientation tensor were subsequently computed using in-house written MATLAB image analysis routines. This was done using a modified version of the Method of Ellipses, a topic which will be discussed in detail in an upcoming publication [33]. In brief, this method consists of application of the traditional Method of Ellipses (MOE) within the end-gated geometry [34]. However, the traditional MOE must be modified to utilize the application of the modified image analysis width determined by Hofmann et al. [35]. This is necessary in the regions of highly shear-aligned fibers near the mold side walls, where too narrow of an image analysis region results in an increase in partial ellipses and associated error. This modified method allows for the accurate determination of the unambiguous components of the orientation tensor at all inspection points within the EGP [36].

5.6 Results

5.6.1 Orientation Tensors

The results section presents three different types of data: predicted orientation values using the rigid fiber model, predicted orientation values using the semi-flexible fiber model and measured values of experimental orientation. The rigid fiber model produces component values of the orientation tensor described by Eq. (5.8) so that when a component is discussed pertaining the rigid fiber model it will be represented by A_{ii} or A_{ij}. The solution of the semi-flexible fiber model produces two orientation tensors, $\underline{\underline{A}}$ and $\underline{\underline{B}}$ that represent the tensors given in Eq. (5.11) and Eq. (5.12). The results presented in this work utilize a combination of the two orientation tensors given by $\underline{\underline{R}}$ which is that "average" end-to-end orientation of a slightly deformed fiber. The results of the semi-flexible fiber orientation model are given through the components of $\underline{\underline{R}}$. Finally, the measured values of experimental fiber orientation are measured using the method described in the previous section. The method of ellipses generates the rigid model orientation tensor, $\underline{\underline{A}}$. But, recent results published by Hofmann et al. describe a technique of verifying experimental long glass fiber orientation through a "top-down" method and concluded that over the length of the fiber in systems such as the end-gated plaque, $\underline{\underline{A}} \approx \underline{\underline{R}}$. Therefore, the experimentally measured orientation data will be referred to by the components of $\underline{\underline{R}}$.

5.6.2 Temperature Effects

The system under study in this work was clearly non-isothermal and, as such, an effort to incorporate those effects was made. Non-isothermal conditions present issues such as a large thermal gradient near the walls. To minimize the effect of the large thermal gradients near the walls a fast fill time was chosen. The fluid filling simulation showed a steep temperature gradient near the walls but, after calculating both the isothermal and non-isothermal orientation predictions, only an 8% - 10% difference was observed between the two no further than 0.12 mm from the wall.

Another issue that can arise from taking into account the temperature effects in the system is the formation of a skin layer near the wall. This is due to the fluid coming into contact with the mold $\left(\Delta T \approx 70K\right)$ and cooling quickly. For the purpose of the study presented here, and because of the fast fill times chosen in this study, the skin formation was neglected.

5.6.3 Determination of Fiber Orientation Parameters

The fiber interaction coefficient, C_I, and the slip parameter, κ, were determined through the use of a sliding plate rheometer and the analysis of transient stress overshoots in the startup of simple shear as prescribed by Ortman et al. [15] and confirmed for orientation predictions in a CGD by Ortman et al. [16] and Meyer et al. [20]. A randomly oriented $\left(\underline{\underline{A}} \approx \frac{1}{3}\underline{\underline{I}}\right)$ long glass fiber sample was sheared at $\dot{\gamma} = 1\ \mathrm{s}^{-1}$ for 120 seconds. The resulting transient stresses were modeling using a modified Dinh-Armstrong stress tensor form. The values for the slip parameter and fiber interaction coefficient calculated through this method are given in Table 5.2. Using these parameters, fiber orientation was predicted in the end-gated plaque with the results from a single position (centerline at 0% mold fill, position (1) in Figure 5.3) shown for different orientation components in Figure 5.4 (a). The results shown in Figure 5.4 (a) clearly suggest that the characterization technique using only simple shear flow may not completely describe the rheological and orientation evolution of a concentrated fiber system and rheological testing in different flows (i.e. extensional flow) may be required to fully understand the suspension.

The fiber interaction coefficient, C_I, and the slip parameter, κ, were then determined through the fitting of experimental data at the 0% fill position along the centerline of the end-gated plaque (position (1) in Figure 5.3). These parameters were fit using a non-linear least squares analysis with the resulting values given in Table 5.2. The results of the predicted fit are shown in Figure 5.4 (b).

5.6.4 LGF Orientation Predictions at 0% of the Mold Fill

Using the conditions obtained from simulating LGF orientation in the gate region, predictions about LGF orientation in the mold cavity could be made. Figure 5.5 shows the results of predicting fiber orientation at 0% of the mold fill along two widths (50% of the width and 90% of the width) of the mold cavity. Figure 5.5 (b) shows the comparison of experimental LGF orientation data with model predictions at 50% of the mold width (sampling position (2) in Figure 5.3). The rigid and semi-flexible fiber models both qualitatively behave in a similar manner predicting a flatter orientation profile through the thickness of the mold cavity forgoing the traditional “shell-core-shell” profile. The rigid fiber model is observed to qualitatively agree with the experimental fiber orientation data but over predicts the R_{33} component of the orientation tensor and under predicts the R_{11} component of the orientation through the majority

of the mold cavity. The semi-flexible fiber model is also observed to agree qualitatively with experimental data and more accurately describes the observed quantitative experimental fiber orientation values.

The results in Figure 5.5 (c) shows the comparison of experimentally observed fiber orientation values with model predictions at 90% of the mold width (sampling position (3) in Figure 5.3). Both models are observed to qualitatively agree with the shape of the fiber orientation distribution through the mold cavity. Quantitatively, both models perform equally as well at predicting the observed fiber orientation with the exception of the top of the mold cavity, where the semi-flexible fiber model is slightly more accurate.

5.6.5 LGF Orientation Predictions at 10% of the Mold Fill

Comparisons of predictions and experimental data were made at 10% of the mold fill at three widths of the mold in Figure 5.6. First, Figure 5.6 (a) shows predicted LGF orientation values using a rigid and semi-flexible fiber model compared to experimentally observed fiber orientation values at 0% of the mold width (sampling position (4) in Figure 5.3). Both models again predict the classic SCS region commonly associated with fiber orientation predictions in thin cavities. The semi-flexible fiber model predictions are observed to more closely agree with experimental orientation values because of the flatter profile across the mold cavity thickness.

The comparison of predicted LGF orientation values and experimentally obtained data at 50% of the mold width are given in Figure 5.6(b) (sampling position (5) in Figure 5.6). The rigid and semi-flexible fiber models both predict flat orientation profiles through the mold cavity that is observed to be in agreement with experimental LGF orientation values. The rigid fiber model quantitatively under predicts the R_{11} component of the orientation tensor and over predicts the R_{33} component of the orientation tensor. The semi-flexible fiber model is observed to more accurately predict the R_{33} component of the experimental fiber orientation tensor but performs almost as poorly as the rigid fiber model at predicting the R_{11} component of the experimental orientation. Furthermore, the rigid fiber model continues to predict a shallow SCS-type profile through the thickness of the cavity while the semi-flexible fiber model does not.

The comparison of experimental data with predicted orientation values of LGF was also performed at 90% of the mold width given in Figure 5.6 (c) (sampling position (6) in Figure 5.3). The rigid fiber model still slightly predicts a SCS region through the cross-section of the mold but in this case it is observed to be both qualitatively and quantitatively inaccurate. The rigid fiber model under predicts the R_{11} component of fiber orientation and over predicts the R_{33} component of fiber orientation when compared to experimental LGF orientation data. Conversely, the semi-flexible fiber model performs encouragingly more accurately reflecting the qualitatively trends observed in the mold cavity. Specifically, the semi-flexible fiber model is observed to accurately predict the experimental LGF orientation values for both the R_{11} and R_{33} components of orientation with the exception of predictions near the bottom wall for the R_{11} component and top wall for the R_{33} component. The semi-flexible fiber model also more accurately predicts the R_{22} component of fiber orientation when compared to experimental LGF orientation in the center of the mold but performs poorly near the walls.

5.6.6 LGF Orientation Predictions at 40% of the Mold Fill

Comparisons of predicted and observed LGF orientation values continued at 40% of the mold fill given by the plots given in Figure 5.7. Figure 5.7 (a) shows the rigid fiber and semi-

flexible fiber model predictions compared to observed experimental LGF orientation values (sampling position (7) in Figure 5.3). Both model predictions again show SCS regions through the mold cavity thickness. The rigid fiber model over predicts the R_{11} component of orientation near the walls of the cavity and under predicts the R_{11} component of orientation in the center of the cavity when compared to observed experimental values of LGF orientation. The rigid fiber model also under predicts the A_{33} component of orientation and over predicts the R_{22} component of orientation through the entire thickness when compared to experimental LGF orientation data.

The rigid and semi-flexible fiber model were compared against experimentally observed LGF orientation at 50% of the mold width given by Figure 5.7 (b) (sampling position (8) in Figure 5.3). The rigid fiber model predicts a shallow SCS orientation profile through the thickness that is qualitatively incorrect when compared to observed experimental LGF orientation values. The semi-flexible fiber performs marginally better by qualitatively predicting a flatter orientation profile through the thickness but quantitatively does not predict orientation values similar to the observed experimental values with the exception of the R_{33} orientation component in the top half of the mold.

At 90% of the mold width comparisons of model predictions against observed experimental orientation values were again performed with the results shown in Figure 5.7 (c) (sampling position (9) in Figure 5.3). The rigid fiber model predicts a SCS profile through the thickness of the mold cavity that is qualitatively in disagreement with experimental LGF orientation values in this system. The semi-flexible model performs encouragingly both predicting the qualitative trends through the mold cavity as well as agreeing quantitatively with observed orientation for this LGF system. The exception is around "-0.4" normalized mold thickness where the R_{22} component of orientation is over predicted and the R_{11} component of orientation is under predicted.

5.6.7 LGF Orientation Predictions at 90% of the Mold Fill

In order to verify the predictions of fiber orientation near the advancing front, experimental LGF orientation values were compared to model predictions at 90% of the mold fill at three widths shown in Figure 5.8. The results in Figure 5.8 (a) show the comparisons of model predictions with observed LGF orientation at 0% of the mold width (sampling position (10) in Figure 5.3). Both the rigid and semi-flexible fiber models perform qualitatively and quantitatively well at this position when compared to experimental data with the exception of the bottom mold wall. Furthermore, both models over predict the experimental R_{22} component of orientation through the mold cavity.

At 50% of the width similar comparisons of predicted and experimentally observed LGF orientation were carried out and given in Figure 5.8 (b) (sampling position (11) in Figure 5.3). Here the rigid fiber model qualitatively captured the wide orientation distribution through the mold cavity and is observed to correctly predict the experimental R_{33} component of orientation through the mold cavity thickness. Conversely, the rigid fiber under predicts the experimental R_{11} component of orientation and over predicts the R_{22} component of orientation through the mold thickness. The semi-flexible fiber model is observed to qualitatively capture the flat orientation profile through the thickness of the cavity. Furthermore, the semi-flexible fiber model quantitatively describes the observed experimental orientation for the R_{11} and R_{33} orientation components but fails to accurately capture the R_{22} orientation component.

Finally, experimentally observed LGF orientation data and model predictions were compared at 90% of the mold width given by Figure 5.8 (c) (sampling position (12) in Figure 5.3). The rigid fiber model again predicts a SCS profile through the mold cavity thickness that at this sampling position which is in disagreement with experimental LGF orientation values. The semi-flexible fiber model more accurately captures the qualitative trends seen in the experimental data and additionally agrees quantitatively with experimentally observed values of the R_{11} and R_{33} orientation components through the entire mold cavity with the except of near the top wall of the mold. The rigid and semi-flexible fiber models are observed to over predict the R_{22} orientation component albeit the semi-flexible fiber model is closer to experimentally observed values of LGF orientation.

5.7 Conclusions

Long glass fiber orientation predictions were compared to experimentally measured values of orientation in a non-isothermal end-gated plaque at a number of positions in the gate and mold cavity. The orientation predictions were made using a decoupled approach for the stress and orientation tensors and included the gate region and the advancing front. The fiber orientation parameters were first obtained through an independent rheological technique where transient stress overshoots from the startup of simple shear were fitted to a stress model incorporating fiber contribution. The long glass fiber orientation predictions were in poor agreement when compared to experimental orientation data suggesting that a more thorough rheological investigation (i.e. behavior in extensional flow fields) may be necessary to characterize the concentrated suspension. Parameters in the rigid and semi-flexible fiber models were then estimated through fitting experimental orientation data at a single position in the mold (along the centerline at the 0% fill position) using a non-linear least squares analysis and then used to predict fiber orientation at all other locations in the mold. Along the centerline of the mold cavity both the rigid and semi-flexible fiber model qualitatively predicted the trends that were seen in the experimentally obtained fiber orientation data but the semi-flexible fiber model predictions more closely agreed with the experimental values of orientation. Away from the centerline of the mold the semi-flexible fiber model performs encouragingly more accurately predicting experimental fiber orientation at every position investigated with the exception of one position where the rigid and semi-flexible fiber models provided similar accuracy. Along the width of the 0% fill line the model predictions and experimental data suggest that the fiber orientation across the gate/mold interface appears to be a function of mold width so that previous simulations assuming a single orientation across the gate/mold interface could provide less accurate orientation predictions in the mold cavity. Furthermore, the semi-flexible fiber model predicted values of orientation that were in greater agreement with experimental values of orientation. This increase in agreement can be attributed to the semi-flexible fiber model including terms for fiber length in the system and the incorporation of long glass fiber flexibility.

5.8 Acknowledgements

The financial support for this work is from Oak Ridge National Laboratories through the Department of Energy Grant No. 4000122683 and the National Science Foundation Grant No. CMMI-0853537 and is gratefully appreciated and acknowledged. The authors also wish to thank the SABIC Plastics Inc. for supplying the neat matrix resin (VERTON® Series) and the 30 wt. % long glass fiber materials. The authors also acknowledge Syed Mazahir and Mark Cieslinski for the fruitful discussions.

5.9 References

[1] D. J. Klingenberg and L. H. Switzer, "Rheology of sheared flexible fiber suspensions via fiber-level simulations," *Journal of Rheology,* vol. 47, pp. 759-778, May-Jun 2003.

[2] J. M. Crosby, "Recent Advances in Thermoplastic Composites," *Advanced Materials & Processes,* vol. 133, pp. 56-59, Mar 1988.

[3] G. B. Jeffery, "The Motion of Ellipsoidal Particles Immersed in a Viscous Fluid," *Proceedings of the Royal Society a-Mathematical Physical and Engineering Sciences,* vol. 100, pp. 161-179, 1922.

[4] F. Folgar and C. L. Tucker, "Orientation Behavior of Rigid Fibers in Concentrated Suspensions," *Journal of Rheology,* vol. 26, pp. 604-604, 1982.

[5] S. Ranganathan and S. G. Advani, "Fiber-Fiber Interactions in Homogeneous Flows of Nondilute Suspensions," *Journal of Rheology,* vol. 35, pp. 1499-1522, Nov 1991.

[6] R. S. Bay and C. L. Tucker, "Fiber Orientation in Simple Injection Moldings .1. Theory and Numerical-Methods," *Polymer Composites,* vol. 13, pp. 317-331, Aug 1992.

[7] R. S. Bay and C. L. Tucker, "Fiber Orientation in Simple Injection Moldings .2. Experimental Results," *Polymer Composites,* vol. 13, pp. 332-341, Aug 1992.

[8] D. G. Baird, A. P. R. Eberle, G. M. Velez-Garcia, and P. Wapperom, "Fiber orientation kinetics of a concentrated short glass fiber suspension in startup of simple shear flow," *Journal of Non-Newtonian Fluid Mechanics,* vol. 165, pp. 110-119, Feb 2010.

[9] M. Sepehr, P. J. Carreau, M. Moan, and G. Ausias, "Rheological properties of short fiber model suspensions," *Journal of Rheology,* vol. 48, pp. 1023-1048, Sep-Oct 2004.

[10] M. Sepehr, G. Ausias, and P. J. Carreau, "Rheological properties of short fiber filled polypropylene in transient shear flow," *Journal of Non-Newtonian Fluid Mechanics,* vol. 123, pp. 19-32, Oct 15 2004.

[11] M. Sepehr, P. J. Carreau, M. Grmela, G. Ausias, and P. G. Lafleur, "Comparison of rheological properties of fiber suspensions with model predictions," *Journal of Polymer Engineering,* vol. 24, pp. 579-610, Nov-Dec 2004.

[12] R. B. Bird, R. C. Armstrong, and O. Hassager, *Dynamics of polymeric liquids. - 1 : Fluid mechanics*, 2.ed. ed. New York,N.Y.: Wiley, 1987.

[13] C. L. Tucker, J. Wang, and J. F. O'Gara, "An objective model for slow orientation kinetics in concentrated fiber suspensions: Theory and rheological evidence," *Journal of Rheology,* vol. 52, pp. 1179-1200, Sep-Oct 2008.

[14] C. L. Tucker and J. H. Phelps, "An anisotropic rotary diffusion model for fiber orientation in short- and long-fiber thermoplastics," *Journal of Non-Newtonian Fluid Mechanics,* vol. 156, pp. 165-176, Feb 2009.

[15] K. Ortman, D. Baird, P. Wapperom, and A. Whittington, "Using startup of steady shear flow in a sliding plate rheometer to determine material parameters for the purpose of predicting long fiber orientation," *Journal of Rheology,* vol. 56, pp. 955-981, Jul 2012.

[16] K. Ortman, D. Baird, P. Wapperom, and A. Aning, "Prediction of fiber orientation in the injection molding of long fiber suspensions," *Polymer Composites,* vol. 33, pp. 1360-1367, 2012.

[17] S. M. Mazahir, G. M. Vélez-García, P. Wapperom, and D. Baird, "Evolution of fibre orientation in radial direction in a center-gated disk: Experiments and simulation," *Composites Part A: Applied Science and Manufacturing,* vol. 51, pp. 108-117, 8// 2013.

[18] K. J. Meyer, J. T. Hofmann, and D. G. Baird, "Prediction of Short Glass Fiber Orientation in the Filling of an End-Gated Plaque," *Composites Part a-Applied Science and Manufacturing,* 2013.

[19] U. Strautins and A. Latz, "Flow-driven orientation dynamics of semiflexible fiber systems," *Rheologica Acta,* vol. 46, pp. 1057-1064, Oct 2007.

[20] K. J. Meyer, J. T. Hofmann, and D. G. Baird, "Initial conditions for simulating glass fiber orientation in the filling of center-gated disks," *Composites Part A: Applied Science and Manufacturing,* vol. 49, pp. 192-202, 6// 2013.

[21] B. N. Nguyen, S. K. Bapanapalli, J. D. Holbery, M. T. Smith, V. Kunc, B. J. Frame*, et al.*, "Fiber length and orientation in long-fiber injection-molded thermoplastics - Part I: Modeling of microstructure and elastic properties," *Journal of Composite Materials,* vol. 42, pp. 1003-1029, May 2008.

[22] B. E. VerWeyst and C. L. Tucker, "Fiber suspensions in complex geometries: Flow/orientation coupling," *Canadian Journal of Chemical Engineering,* vol. 80, pp. 1093-1106, Dec 2002.

[23] G. M. Velez-Garcia, P. Wapperom, V. Kunc, D. G. Baird, and A. Zink-Sharp, "Sample preparation and image acquisition using optical-reflective microscopy in the measurement of fiber orientation in thermoplastic composites," *Journal of Microscopy,* vol. 248, pp. 23-33, Oct 2012.

[24] S. M. Dinh and R. C. Armstrong, "A Rheological Equation of State for Semiconcentrated Fiber Suspensions," *Journal of Rheology,* vol. 28, pp. 207-227, 1984.

[25] S. G. Advani and C. L. Tucker, "The Use of Tensors to Describe and Predict Fiber Orientation in Short Fiber Composites," *Journal of Rheology,* vol. 31, pp. 751-784, Nov 1987.

[26] D. H. Chung and T. H. Kwon, "Fiber orientation in the processing of polymer composites," *Korea-Australia Rheology Journal,* vol. 14, pp. 175-188, Dec 2002.

[27] D. H. Chung and T. H. Kwon, "Invariant-based optimal fitting closure approximation for the numerical prediction of flow-induced fiber orientation," *Journal of Rheology,* vol. 46, pp. 169-194, Jan-Feb 2002.

[28] G. M. Velez-Garcia, S. M. Mazahir, P. Wapperom, and D. G. Baird, "Simulation of Injection Molding Using a Model with Delayed Fiber Orientation," *International Polymer Processing,* vol. 26, pp. 331-339, Jul 2011.

[29] L. F. R. Shampine, M.W., "The MATLAB ODE Suite," ed. 24 Prime Park Way, Natick, MA 01760: The Mathworks, Inc.

[30] B. N. Nguyen, S. K. Bapanapalli, J. D. Holbery, M. T. Smith, V. Kunc, B. J. Frame*, et al.*, "Fiber Length and Orientation in Long-Fiber Injection-Molded Thermoplastics – Part I: Modeling of Microstructure and Elastic Properties," *Journal of Composite Materials,* vol. 42, pp. 1003-27, 2008.

[31] S. G. Advani and E. M. Sozer, *Process modeling in composites manufacturing*, 2nd ed. Boca Raton, FL: CRC Press, 2011.

[32] G. Velez-Garcia, P. Wapperom, V. Kunc, D. Baird, and A. Zink-Sharp, "Sample preparation and image acquisition using optical−reflective microscopy in the measurement of fiber orientation in thermoplastic composites," *Journal of Microscopy,* vol. 248, pp. 23-33, 2012.

[33] J. T. Hofmann, K. J. Meyer, and D. G. Baird, 2013 (In Preparation).

[34] P. J. Hine, N. Davidson, R. A. Duckett, and I. M. Ward, "Measuring the Fiber Orientation and Modelling the Elastic Properties of Injection-moulded Long-glass-fiber-reinforced Nylon," *Composites Science and Technology,* vol. 53, pp. 125-131, 15 December 1994 1994.

[35] J. T. Hofmann, G. M. Vélez−Garcia, D. G. Baird, and A. R. Whittington, "Application and evaluation of the method of ellipses for measuring the orientation of long, semi−flexible fibers," *POLYMER COMPOSITES,* 2013.

[36] G. M. Vélez-García, P. Wapperom, D. G. Baird, A. O. Aning, and V. Kunc, "Unambiguous orientation in short fiber composites over small sampling area in a center-gated disk," *Composites Part A: Applied Science and Manufacturing,* vol. 43, pp. 104-113, 2012.

5.10 Figures

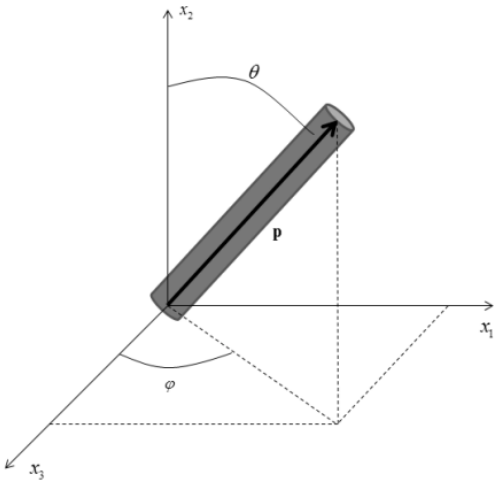

Figure 5.1: Vector definition of rigid-fiber. The fiber is characterized by the vector, $\underline{p}$, as well as the azimuthal and zenith angles, ϕ and θ respectively.

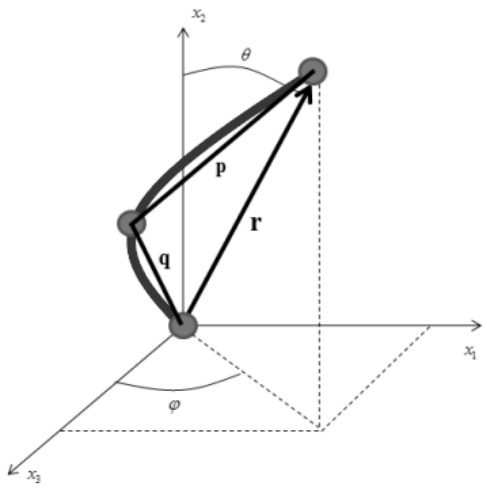

Figure 5.2: Vector definition of flexible fiber. The fiber is characterized by the vectors, $\underline{p}$ and $\underline{q}$, as well as the azimuthal and zenith angles, ϕ and θ respectively. The two vector are of equal length, l_b, and there is an internal resistivity to bending, k. The end-to-end vector, $\overline{r}$, is also defined to determine a fiber's "average" direction.

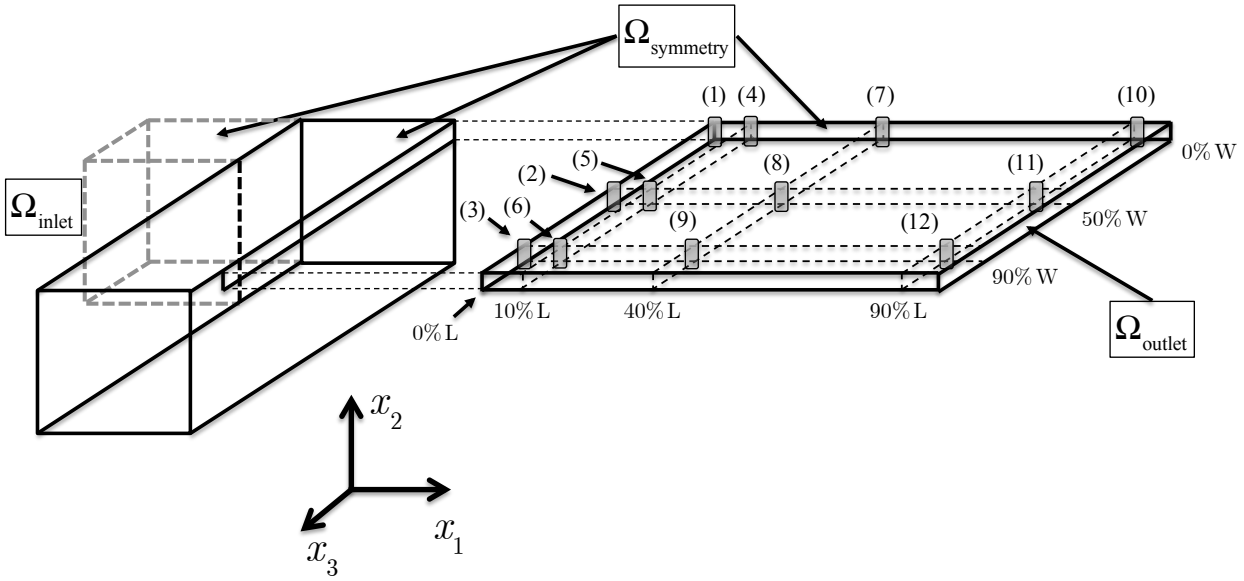

Figure 5.3: End-gated plaque geometry with highlighted regions of interest (1) 0% fill, 0% width, (4) 10% fill, 0% width, (7) 40% fill, 0% width, (10) 90% fill, 0% width, (2) 0% fill, 50% width, (5) 10% fill, 50% width, (8) 40% fill, 50% width, (11) 90% fill, 50% width, (3) 0% fill, 90% width, (6) 10% fill, 90% width, (9) 40% fill, 90% width, (12) 90% fill, 90% width.

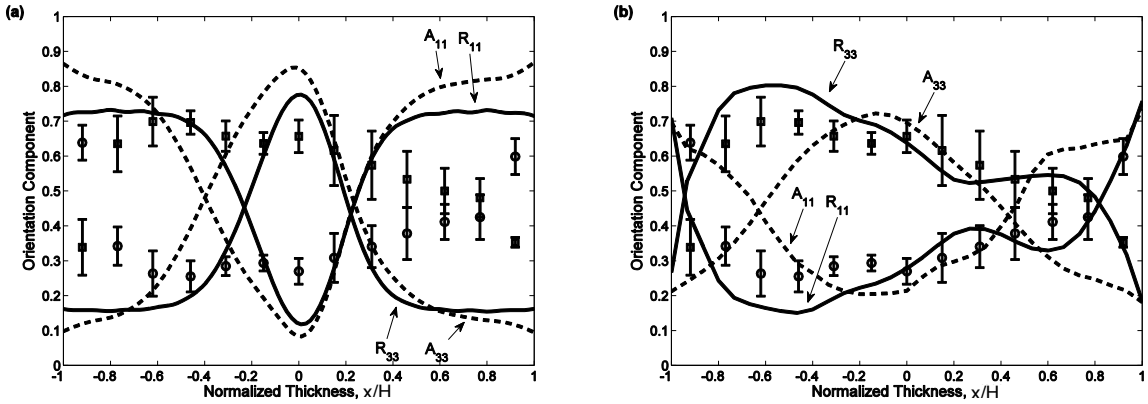

Figure 5.4: Orientation predictions using the (a) rheologically determined parameters (rigid = dashed $\kappa^{R} = 0.23$ and $C_I^{R} = 0.005$, semi-flexible = solid, $\kappa^{SF} = 0.13$ and $C_I^{SF} = 0.053$) and (b) experimentally fit parameters (rigid = dashed $\kappa = 0.0087$ and $C_I = 0.2278$, semi-flexible = solid, $\kappa = 0.0039$ and $C_I = 0.4843$). Predictions are compared to experimentally determined fiber orientation (A_{11} - o, A_{33} - □) at 0% of the mold width and 0% mold fill (position (4) in Figure 5.3).

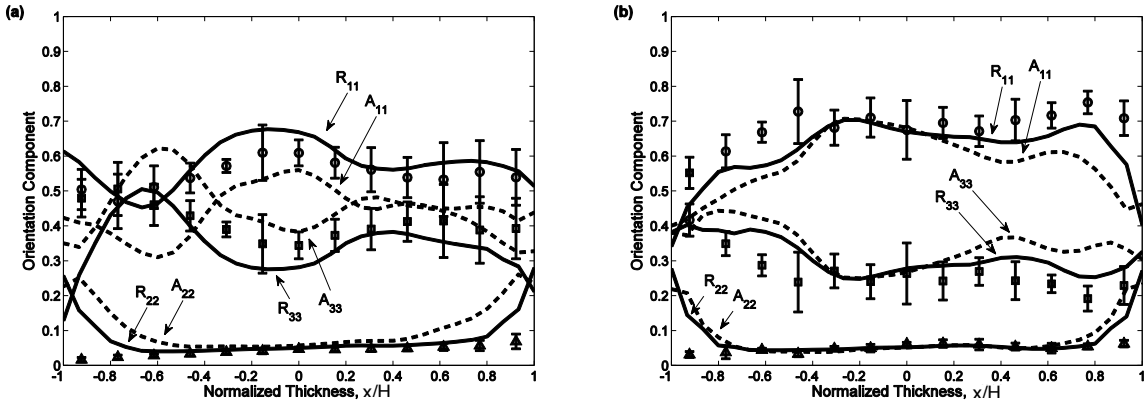

Figure 5.5: Model predictions of rigid (dashed) and semi-flexible (rigid) models compared to experimentally determined fiber orientation (A_{11} - o, A_{22} - Δ, A_{33} - □) at 0% of the mold fill at (a) 0% mold width (position (4) in), (b) 50% mold width (position (5) in Figure 5.3) and (c) 90% mold width (position (6) in Figure 5.3).

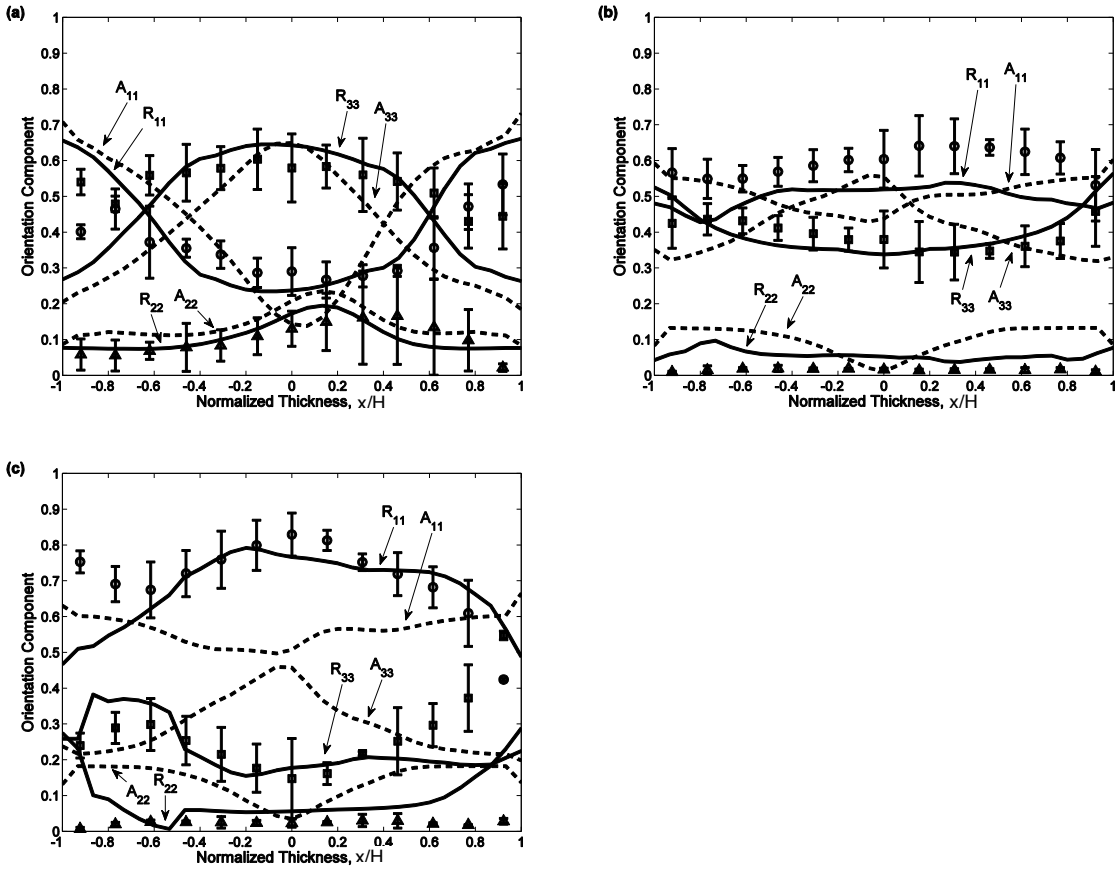

Figure 5.6: Model predictions of rigid (dashed) and semi-flexible (solid) models compared to experimentally determined fiber orientation (A_{11} - o, A_{22} - Δ, A_{33} - □) at 10% of the mold fill at (a) 0% mold width (position (7) in Figure 5.3), (b) 50% mold width (position (8) in Figure 5.3) and (c) 90% mold width (position (9) in Figure 5.3).

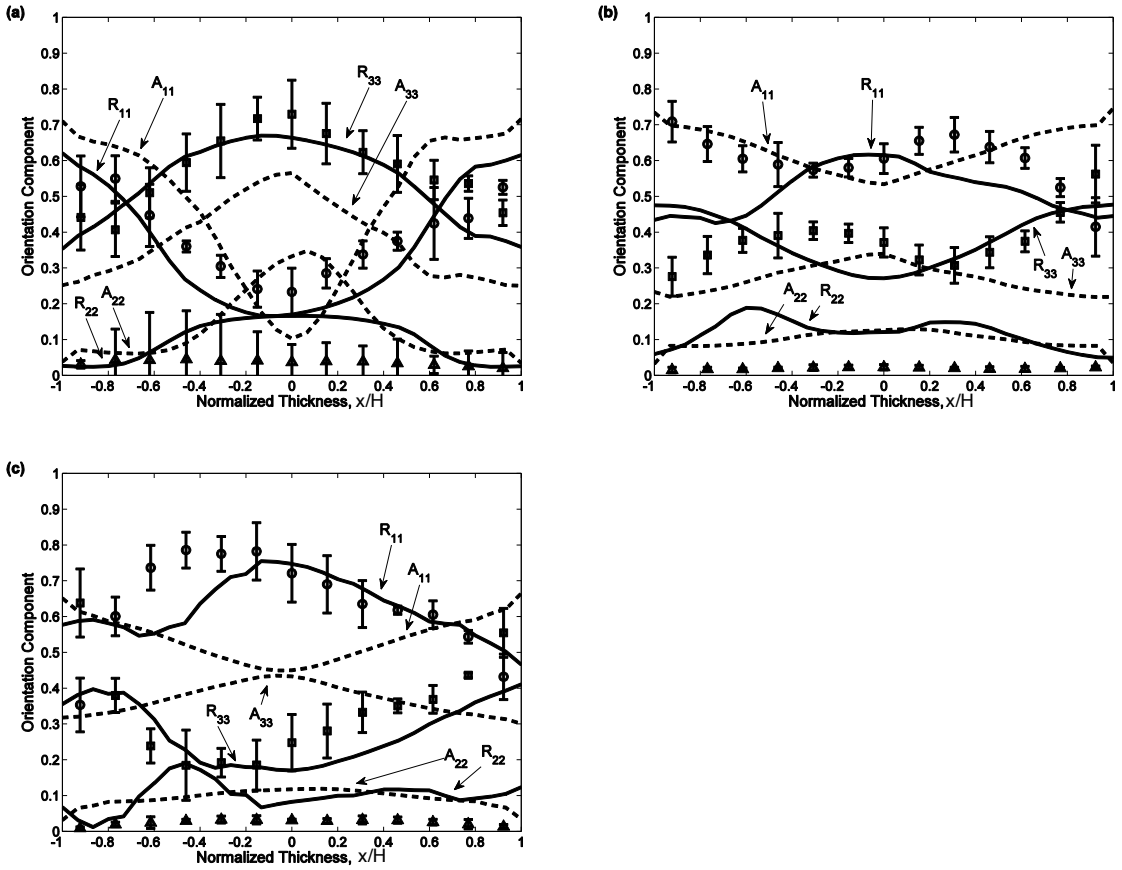

Figure 5.7: Model predictions of rigid (dashed) and semi-flexible (solid) models compared to experimentally determined fiber orientation (A_{11} - o, A_{22} - Δ, A_{33} - □) at 40% of the mold fill at (a) 0% mold width (position (10) in Figure 5.3), (b) 50% mold width (position (11) in Figure 5.3) and (c) 90% mold width (position (12) in Figure 5.3).

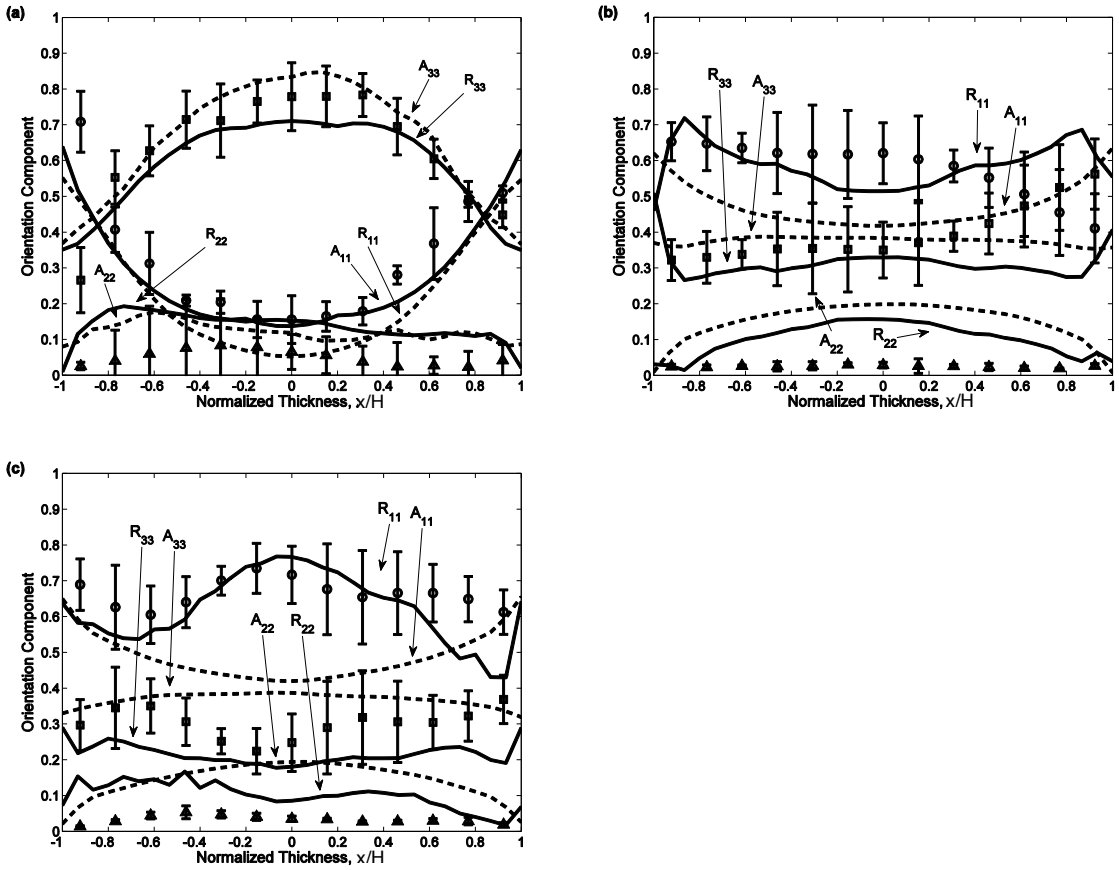

Figure 5.8: Model predictions of rigid (dashed) and semi-flexible (solid) models compared to experimentally determined fiber orientation (A_{11} - o, A_{22} - Δ, A_{33} - □) at 90% of the mold fill at (a) 0% mold width (position (13) in Figure 5.3), (b) 50% mold width (position (14) in Figure 5.3) and (c) 90% mold width (position (15) in Figure 5.3).

Parameter	Value
$\rho\left(\mathrm{kg/m^3}\right)$	1100
$\eta_0\left(\mathrm{Pa\cdot s}\right)$	227.0
$\lambda\left(\mathrm{s^{-1}}\right)$	0.0065
n	0.2759
a	0.8263
$C_P\left(\mathrm{J/kg\cdot K}\right)$	2620
$k\left(\mathrm{W/m\cdot K}\right)$	0.3100
$\alpha\left(\mathrm{K}\right)$	4937
$T_\alpha\left(\mathrm{K}\right)$	452.7

Table 5.1: Material properties for non-isothermal finite element simulation obtained from rheological testing in torsional rheometer (RMS-800, Rheometrics Inc.) and extensional rheometer (Rheograph 2001, Göttfert Inc.).

Rheological Technique Fit		
Parameter	**Rigid Model**	**Flexible Model**
κ	0.2500	0.1300
C_I	0.0050	0.0530
Experimental Data Fit		
Parameter	**Rigid Model**	**Flexible Model**
κ	0.0087	0.0039
C_I	0.2278	0.4843

Table 5.2: Fiber orientation simulation parameters.

Chapter 6

Conclusions

Preface

This section addressed the success or failure of the proposed research objectives and general conclusions linked to those objectives.

6 Conclusions

The following conclusions are made based on the research objectives described in Chapter 1 along with an additional conclusions that were found as a result of working on the objectives presented in Chapter 1:

1. Utilizing the sprue, gate and mold simulation method along with a using a semi-flexible fiber model showed an improved agreement between predicted and experimentally obtained values of long glass fiber orientation in a center-gated disk than of using the semi-flexible model in Hele-Shaw flow.

   The SGM method was used to simulate the sprue, gate and mold region of a center-gated injection molding test geometry including the fountain flow effect to assess if a semi-flexible fiber model was more accurate at predicting experimental long glass fiber orientation values than previous work utilizing the semi-flexible fiber model in Hele-Shaw flow. The semi-flexible fiber model was observed to more accurately predict long glass fiber orientation than the rigid fiber model when compared to experimentally measured orientation data at a number of positions in the center-gated disk. Furthermore, it was shown that the SGM method of simulation produced entry conditions of fiber orientation to the mold that were very similar to the asymmetric experimental orientation values obtained suggesting that assuming a "random" or "planar random" orientation initial condition at the gate/mold interface may not be sufficient for accurate fiber orientation predictions. Rheologically determined orientation parameters were used in the center-gated disk long fiber orientation predictions.

2. The extension of the sprue, gate and mold method to a three dimensional end-gated plaque geometry showed that the strain reduction factor model can predict short glass fiber orientation values that are in agreement with experimentally determined values of orientation. Furthermore, it was observed that objectivity did not play a role in the prediction of short glass fiber orientation at the positions investigated in the mold cavity.

   The sprue, gate and mold of a three-dimensional end-gated plaque was incorporated into a single domain to predict short glass fiber orientation using the strain reduction factor model (SRF) and the reduced strain closure model (RSC) [1, 2]. A number of outcomes from this work are relevant to the proposed research objective. First, the RSC and SRF models provide similar results at all of the locations investigated in the end-gated plaque mold cavity suggesting that objectivity may not play a role in more complicated molding geometries. Second, the SRF model does a reasonably good job at predicting fiber orientation in the end-gated plaque mold at a number of the positions investigated when compared to experimentally measured values of short glass fiber orientation. Finally, the assumption of a fiber orientation as initial conditions at the gate/mold interface (i.e. "random" or "planar random") is most likely incorrect. This is clearly discerned by looking at both the predicted and experimentally measured orientation values that suggest an inlet condition dependence of mold thickness as well as mold width.

   On the first attempt at prediction short glass fiber orientation, orientation parameters obtained from fitting transient stress overshoots were used but over-predicted the evolution rate of orientation in the system. As a secondary approach, the orientation

parameters were fit to a single position in the end-gated mold cavity and then used to predict orientation at all other locations of interest.

3. The sprue, gate and mold method was applied to predict long glass fiber orientation in the end-gated plaque and showed that the semi-flexible fiber model can accurately predict experimentally determined values of fiber orientation both along and away from the centerline of the geometry.

   The sprue, gate and mold method was again used to make predictions both along and away from the centerline of the end-gated plaque mold region. It was observed that the semi-flexible fiber model, using the sprue, gate and mold method, produced orientation predictions that more closely agreed with experimentally measured values of long glass fiber orientation at a majority of the points investigated in the mold cavity suggesting that the work here has significantly improved currently predictive techniques for longer fiber systems. But, at a few locations predictions of long glass fiber orientation did not agree well with measured values of long glass fiber orientation suggesting that further study of concentrated long fiber suspensions is necessary.

   On the first attempt at predicting long glass fiber orientation, orientation parameters obtained from fitting transient shear stress overshoots generated in the sliding plate rheometer were used but over-predicted the evolution rate of orientation in the system. As a secondary approach, the orientation parameters were fit to a single position in the end-gated mold cavity and then used to predict orientation at all other locations of interest.

4. The technique of using a sliding plate rheometer and a simple shear field to determine orientation parameters failed to produce accurate orientation predictions in the end-gated plaque.

   During the course of working on the proposed research objectives for the end-gated plaque geometry an issue arose with the ability to obtain orientation parameters from sliding plate tests using simple shear given by Ortman et al. [3]. For the center-gated disk geometry, orientation parameters used in the prediction of glass fiber orientation were obtained through fitting transient stress growth overshoots of the concentrated suspension in simple shear flow. Reasonable agreement was found between the predicted values of glass fiber orientation and observed experimental orientation values. But, in the end-gated plaque, the same technique for determining the orientation parameters yielded poor short and long glass fiber predictions at all of the positions investigated. This led to the adoption of a non-linear least squares fitting technique for the three diagonal orientation components at one position in the end-gated mold. The parameters fitted to experimental orientation data were an order of magnitude different (smaller for the slip parameter, larger for the interaction coefficient) than the parameters obtained by fitting the stress tensor to the transient stress growth in simple shear suggesting that testing of concentrated suspensions in other simple flows (i.e. extensional flow) may be required to characterized these complex suspensions through rheology alone.

6.1 References

[1] M. Sepehr, G. Ausias, and P. J. Carreau, "Rheological properties of short fiber filled polypropylene in transient shear flow," *Journal of Non-Newtonian Fluid Mechanics,* vol. 123, pp. 19-32, Oct 15 2004.

[2] J. Wang, J. F. O'Gara, and C. L. Tucker, "An objective model for slow orientation kinetics in concentrated fiber suspensions: Theory and rheological evidence," *Journal of Rheology,* vol. 52, pp. 1179-1200, Sep-Oct 2008.

[3] K. Ortman, D. Baird, P. Wapperom, and A. Whittington, "Using startup of steady shear flow in a sliding plate rheometer to determine material parameters for the purpose of predicting long fiber orientation," *Journal of Rheology,* vol. 56, pp. 955-981, Jul 2012.

Chapter 7

Recommendations

Preface

This section details some suggested improvements that can be made using this project as a basis.

7 Recommendations

The following recommendations are based on overall conclusions drawn from the accomplishment of the research objectives as well as overall concerns with the work.

1. Rewrite the fiber orientation software in the format of an ANSYS "user defined function."

   All of the numerical predictions of short and long glass fiber orientation presented in this work relied on the use of a combination of the finite element method as well as the finite difference method. This approach provided numerical orientation predictions that were in agreement with experimentally determined orientation values in the center-gated disk and end-gated plaque geometry. One major drawback to the approach used in this work is that the finite difference method is difficult to extend to more complex geometric shapes including curves and multiple contractions and expansions as well as multiple gates.

   The finite element modeler used for the prediction of the fluid entering the mold cavity, ANSYS Polyflow, is able to solve a "user defined function" or C code which has been written for an explicit set of equations. It would be of great value to rewrite the fiber orientation equations in a form that ANSYS Polyflow is able to solve simultaneously with the flow and heat equations.

2. Determine a more accurate way to represent a single fiber's flexibility.

   Previous efforts used a simply supported beam with a point load in the center (shown in Figure 7.1) to estimate the fiber flexibility parameter, k.

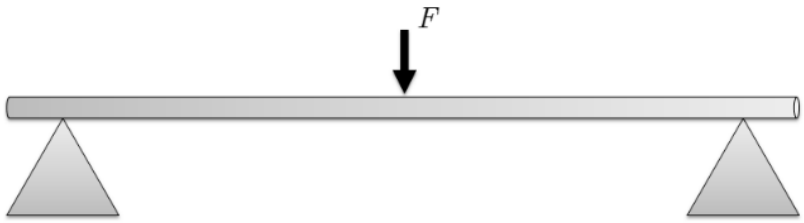

Figure 7.1: Qualitative representation of simply supported beam with a point force acting on the center.

To calculate the fiber flexibility, Eq. (7.1) was proposed:

$$k_i = \left(\frac{E_Y}{64\eta_m} \right) \frac{d^3}{l_b^3} \tag{7.1}$$

In Eq. (7.1) k_i is a flexibility for a particular fiber length, E_Y is the Young's modulus, η_m is the matrix viscosity, d is the diameter of the individual fiber and l_{b_i} is the half-length of a particular fiber length in the distribution. The fiber flexibility parameter is then averaged using either a weight- or number-averaged technique.

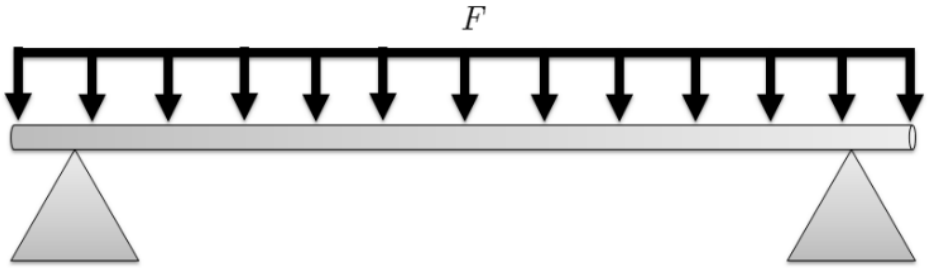

Figure 7.2: Qualitative representation of simply supported beam with a distributed force acting on the length of the beam.

One improvement to this predictive technique would be to apply a distributed load over the length of the fiber which would more accurately reflect the fiber-fluid interaction in the suspension. This could be done by again applying basic mechanics techniques but instead derive the relation for a distributed load over the fiber.

3. Determine the sensitivity of the semi-flexible fiber model with respect to the fiber length.

Currently the fiber flexibility parameter, k, is estimated in the orientation predictions (Eq. (7.1)) but the effect that these parameters play in dictating orientation evolution is still not well understood.

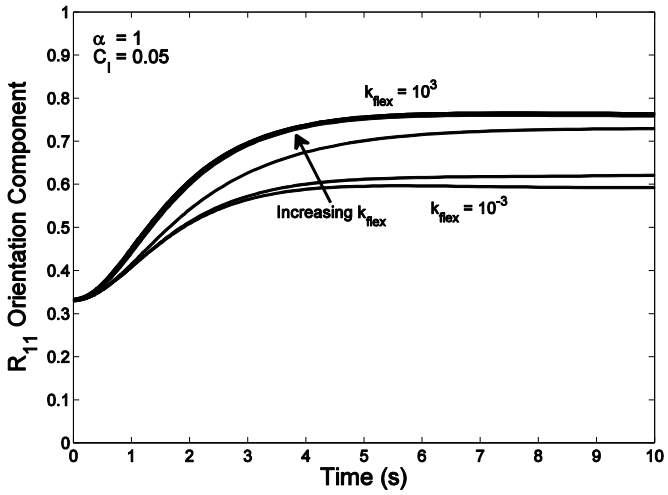

Figure 7.3: Predicted R_{11} orientation component in simple shear flow for increasing values of the flexibility parameter, k.

The results in Figure 7.3 show the effect of the flexibility parameter on the prediction of the R_{11} component of orientation when in the startup of simple shear. It is observed that reducing the value of the flexibility parameter (thereby increasing the flexibility of the fiber) can slow the evolution of orientation as well as affect the steady-state value of orientation. Experimental studies using different fibers with different values of the flexibility parameters would be useful in confirming the effect seen in Figure 7.3.

The value of the length parameter, l_b, can also play a role in dictating the evolution of orientation within a fiber suspension in two ways: the length parameter

directly weights one term in the semi-flexible fiber orientation equations and it is also used in the estimation of the fiber flexibility parameter, k. Currently the length term has been determined for one glass fiber system (the SABIC glass fiber/polypropylene material investigated in Chapter 3 and Chapter 5 of this work) so much could be learned from simulating materials with different fiber lengths. Furthermore, the length parameter is treated as a constant value in the predictions presented throughout this work but in reality the fiber length of a system is represented by a distribution. Incorporating this distribution into the orientation predictions or possibly even a time-dependent fiber breakage model could further increase the accuracy of the orientation simulations.

4. Determine a more robust method for determining the orientation parameters.

The work presented here attempts to provide a foundation for the prediction of short and long glass fiber orientation in complex test geometries using orientation equation parameters obtained from transient stress growth experiments performed in a sliding plate rheometer. The prediction of long glass fiber orientation for a center-gated disk geometry used orientation parameters obtained from the simple shear flow tests and found good agreement with experimental data using a semi-flexible fiber model. This suggested that the orientation parameters for a particular suspension could be estimated through stress growth experiments in simple shear alone. The same orientation parameters for the composite materials were used in the prediction of long glass fiber orientation in the end gated plaque with significantly less accurate results and instead a fitting method was adopted to obtain agreement with predicted and experimental results. This discrepancy may suggest that simple shear may not be sufficient to determine the orientation parameters for predictions and testing in other simple flow fields (i.e. extensional flow) may be necessary. Furthermore, a more accurate stress tensor model may be necessary to describe these very complex concentrated fiber suspensions.

Appendix: Simulation Software Overview

A Appendix: Simulation Software Overview

Fiber Orientation Simulation Software Overview

Kevin J. Meyer[1]

Polymer Processing Laboratory

Department of Chemical Engineering, Virginia Tech

A.1 Introduction

The purpose of this document is to provide an end-user with enough basic knowledge about the fiber orientation prediction method so that he/she is proficient in using the software and numerical method without too much background. Furthermore, it may be necessary to edit functions from time to time and some of the latter material will go into detail on the numerical methods so that this may also be possible for an advanced user.

During the development of this numerical package, a few assumptions were made at the outset. To use the code as it was originally designed the same assumptions must be made about the systems that are being simulated (carbon fiber, liquid crystalline polymer, etc.). The assumptions are as follows:

- The fiber orientation equations are based on rigid rod theory of polymers and as such, are treated as either rigid or semi-rigid rods during the simulations. If the system under study cannot be assumed rigid or close to rigid the equations for fiber orientation may not produce viable results [1].
- The method employed for simulation of the rigid or semi-rigid rods is a continuum method so that "orientation tensors" can be used. If a molecular simulation or some other non-continuum based simulation would better suit the system under study this method may not produce viable results [2].
- Finally, the use of the two step method given here (finite element to calculate velocity profiles and finite difference to calculate orientation profiles) is only accurate if the stress and orientation equations may be decoupled [3, 4]. A coupled analysis is possible through the use of a user-defined function within Polyflow but, at the date of this publication, has not been completed.

A.2 Finite Element Analysis for Injection Molding Simulations

The following section will outline the details for using finite element analysis (FEM) for solving the injection molding problems currently encountered in the polymer processing laboratory. This section is not meant to demonstrate the full power of the FEM inside the Polyflow environment but is instead a guide for those only concerned with injection molding simulations.

[1] Kevin J. Meyer (kjmeyer@vt.edu)
Department of Chemical Engineering
156 Randolph Hall
Virginia Tech, Blacksburg, VA 24061
Phone: +1 941 916 0725.

A.2.1 Steady-State Simulation

The first run of any type of simulation should be a steady-state form of the more complicated problem. The steady-state form will likely yield insight to any major problems that can occur during the more complex simulation which may include high Reynolds numbers, high Péclet numbers, high Brinkman numbers, a combination of all three or other numerical instabilities.

A.2.1.1 *Isothermal Laminar Flow Problem*

The first step in the solution of the mold-filling problem is the isothermal steady-state solution. The flow is assumed laminar and incompressible so that the continuity equation and the Cauchy momentum equation can be written as (A.1) and (A.2) where $\overline{v}$ is the velocity vector and $\overline{\overline{\sigma}}$ is the total stress tensor[2]:

$$\nabla \cdot \overline{v} = 0 \tag{A.1}$$

$$\nabla \cdot \overline{\overline{\sigma}} = 0 \tag{A.2}$$

The total stress tensor requires the specification of an extra stress tensor for the fluid. In the fiber orientation simulations the decoupled approach was taken and the fiber loading fraction was taken into account through the phenomenological parameters in the fiber orientation equations [3, 5-7]. Therefore, the fluid was treated as a generalized Newtonian fluid given by (A.3)[3] where $\eta(\dot{\gamma})$ is any relation for the viscosity as a function of shear rate and $\overline{\overline{\dot{\gamma}}}$ is the rate of strain tensor:

$$\overline{\overline{\tau}} = \eta(\dot{\gamma})\overline{\overline{\dot{\gamma}}} \tag{A.3}$$

At this point any empiricism can be chosen to represent the viscosity of the fluid but generally speaking the Carreau-Yasuda model fits many shear-thinning materials and converges quickly[4] where η_0 is the zero shear viscosity, η_∞ is the infinite shear viscosity, λ governs the onset of shear thinning and n and a govern the degree of shear thinning.

$$\frac{\eta(\dot{\gamma}) - \eta_\infty}{\eta_0 - \eta_\infty} = \left[1 + \left(\lambda|\dot{\gamma}|\right)^a\right]^{\frac{n-1}{a}} \tag{A.4}$$

The simplified solution will help determine how accurate or inaccurate the mesh is and where problems may arise during the solution. Often this simulation will be run a number of times moving back and forth between this and mesh generation to determine the optimal mesh characteristics. An example of this is shown below comparing the mesh revision number versus computational time.

---

[2] ANSYS Polyflow allows for a number of interpolation methods in steady state but the researcher is strongly encouraged to use only linear interpolations for velocity and pressure as linear interpolation is the highest allowed order in the transient simulations.

[3] ANSYS Polyflow has the ability to handle a number of stress tensor representations including generalized Newtonian fluids, differential viscoelastic models and simplified integral viscoelastic models. Convergence and numerical stability is directly related to the complexity of the model.

[4] ANSYS Polyflow has a wide variety of GNF models including Power Law, Bird-Carreau, Carreau-Yasuda, Cross, Bingham, Herschel-Bulkley, etc.

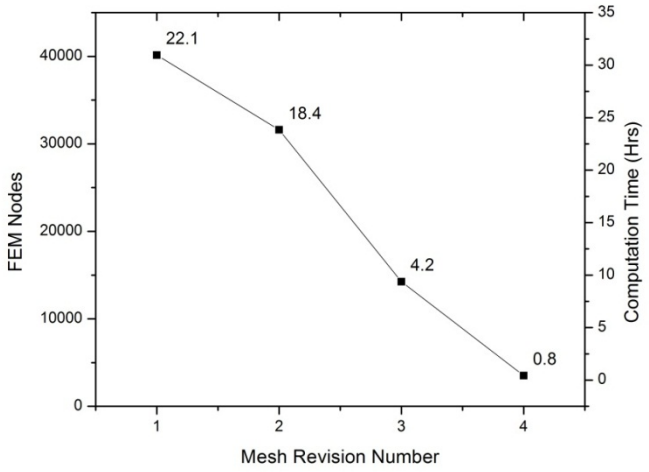

Figure A.1: Computational Time and FEM Nodes vs. Number of Refinements for center-gated disk. Revision 1: tetrahedral meshing only, automated meshing from ANSYS meshing utility. Revision 2: Quad/Tet meshing combination, manual meshing. Revision 3: hexagonal dominant meshing, automated meshing. Revision 4: Hexagonal meshing only, manual meshing.

A.2.1.2 *Non-Isothermal Laminar Flow Problem*

The next step in the solution of the mold filling problem is to solve the same problem as presented in § A.2.1.1 with the addition of temperature. The heat equation including the viscous dissipation term is given in (A.5) where ρ is the density of the fluid, C_P is the heat capacity of the fluid, k is the thermal conductivity of the fluid, $\overline{\overline{\tau}}$ is the extra stress tensor and $\overline{\overline{\dot{\gamma}}}$ is the rate of strain tensor:

$$\rho C_P \frac{DT}{Dt} = k\nabla^2 T + \overline{\overline{\tau}} : \overline{\overline{\dot{\gamma}}} \tag{A.5}$$

The FEM simulations should also take into account the viscosity's dependence on temperature. The Polyflow environment takes this viscosity enhancement into account through (A.6):

$$\eta(\dot{\gamma}, T) = \eta\left(\dot{\gamma} \cdot H(T)\right) H(T) \tag{A.6}$$

Where $H(T)$ is an Arrhenius-type relationship given by (A.7)[5,6]:

$$H(T) = \exp\left[\frac{\alpha}{T - T_0} - \frac{\alpha}{T_\alpha - T_0}\right] \tag{A.7}$$

[5] ANSYS Polyflow has different ways of representing temperature dependence of viscosity including Arrhenius, WLF and Fulcher methods.

[6] It is often necessary to use Picard iterations on the viscosity and increase the number of iterations for convergence.

A.2.2 Volume of Fluid Approach for Transient Filling Simulation

Once the steady-state problem has been solved, the next step is to solve the time dependent flow problem. This involves the use of a time-marching iterative method where the fluid is tracked on the domain as it fills the mold cavity. The following section is an outline of how the time-dependent simulation is solved.

A.2.2.1 *The Volume of Fluid Method*

The volume of fluid (VOF) method was used inside the ANSYS Polyflow environment to simulate the transient mold filling process [8]. The VOF method solves the transport equation, given by (A.8), for the volume fraction of fluid at each node where ϕ is the volume fraction of the liquid phase and $\overline{v}$ is the velocity vector at that node.

$$\frac{\partial \phi}{\partial t} + \overline{v} \cdot \nabla \phi = 0 \tag{A.8}$$

The solution of this equation allows for the fluid front to be tracked as it moves through the empty cavity thereby incorporating the "fountain flow" effect in the simulations. The influence of this complex moving front has been shown to impact fiber orientation predictions [4, 9, 10] in axisymmetric radially diverging flow.

An advantage that the Polyflow solver has over many other VOF solvers is that Polyflow does not solve the "two-fluid" problem commonly associated with the VOF method (liquid entering the mold and air exiting the mold simultaneously). The calculations presented here only calculate the velocity of the fluid in the mold at any node where the volume fraction, ϕ, is greater than zero thereby decreasing the computational time necessary to solve the mold-filling problem.

Assumptions Involved with Using the Volume of Fluid Method:

- Linear interpolation across elements.
- Streamline upwinding to solve transport equation.
- Transport equation is decoupled from motion equations and solved at each time step.
- Transport equation uses sub-element interpolation.
- Time step is automatically chosen based on internal calculations performed by Polyflow.

A.2.2.2 *Evolution Parameters for Non-Isothermal Solution Stability*

During the solution of the non-isothermal time-dependent flow simulation high Brinkman numbers (viscous heating/conduction) can be encountered and are dealt with using an evolution scheme on certain material parameters.

$$\overline{V}_{IN} = \begin{cases} t_i \le \frac{1}{2} t_{\text{fill}} & \overline{V}_{IN}\, t_i \\ t_i > \frac{1}{2} t_{\text{fill}} & \overline{V}_{IN} \end{cases} \tag{A.9}$$

$$C_P = \begin{cases} t_i \le \frac{1}{2} t_{\text{fill}} & C_P\, t_i \\ t_i > \frac{1}{2} t_{\text{fill}} & C_P \end{cases} \tag{A.10}$$

$$k = \begin{cases} t_i \le \frac{1}{2} t_{\text{fill}} & k/t_i \\ t_i > \frac{1}{2} t_{\text{fill}} & k \end{cases} \tag{A.11}$$

$$F\left(\overline{\overline{\tau}} : \overline{\overline{\dot{\gamma}}}\right) = \begin{cases} t_i \le t_{\text{fill}} & F\, t_i \\ t_i > t_{\text{fill}} & F \end{cases} \tag{A.12}$$

A.3 Finite Difference Calculations for Orientation Predictions

The bulk of my thesis has used the computations that will be presented in this section. As such, great detail will be given whenever possible to help the read understand not only the author's method of solution for a specific problem but also the numerics involved with the problem.

A.3.1 Probability Distribution Function and Orientation Tensors

All of the models in the software package accompanying this text used the concept of an orientation tensor described by Advani and Tucker [2]. An orientation tensor is a compact way of representing a rigid rod's orientation instead of using either orientation parameters, which involve some assumption about the orientation state [11, 12], or the orientation distribution function, which can become computationally inefficient.

The orientation state of a given rod at a point in space can be described by the probability distribution function (PDF) $\psi(\theta_1, \phi_1)$. This function is defined so that the probability of finding a fiber between the angles of (θ_1, ϕ_1) and $(\theta_1 + d\theta, \phi_1 + d\phi)$ is unity.

$$P(\theta_1 \le \theta \le \theta_1 + d\theta, \phi_1 \le \phi \le \phi_1 + d\phi) = \psi(\theta_1, \phi_1) \sin\theta \, d\theta \, d\phi \tag{A.13}$$

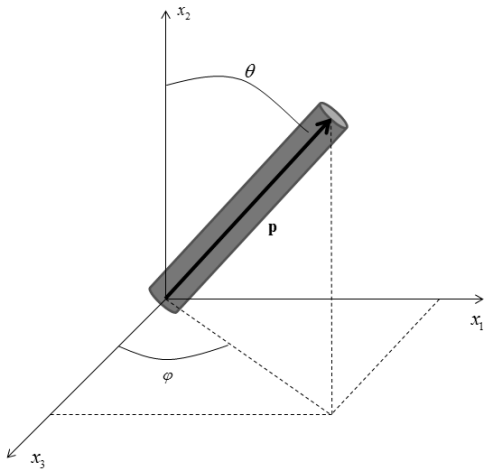

Figure A.2: Vector definition of rigid-fiber. The fiber is characterized by the vector, $\overline{p}$, as well as the azimuthal and zenith angles, ϕ and θ respectively.

We may instead describe an equivalent orientation of a rod as a unit vector, $\overline{p}$, (shown in Figure A.2) and may write the PDF in terms of that vector, $\psi(\overline{p})$. The components of the vector of the

rod, $\bar{p}$, are given by (A.14)-(A.16) and are interrelated since the length of the vector is fixed $\left(p_i p_j = 1\right)$:

$$p_1 = \sin\theta\cos\phi \tag{A.14}$$

$$p_2 = \sin\phi\sin\theta \tag{A.15}$$

$$p_3 = \cos\theta \tag{A.16}$$

The PDF, $\psi(\theta,\phi)$, must satisfy certain conditions so a continuum form must also satisfy these same conditions. First a fiber at any angle (θ,ϕ) must be indistinguishable from a fiber oriented at an angle $(\pi-\theta,\phi+\pi)$ so that ψ is periodic or $\psi(\bar{p}) = \psi(-\bar{p})$. Second, the PDF must be normalized given by (A.17):

$$\int_{\theta=0}^{\pi}\int_{\phi=0}^{2\pi}\psi(\theta,\phi)\sin\theta\, d\theta\, d\phi = \oint\psi(\bar{p})\, d\bar{p} = 1 \tag{A.17}$$

The third condition describes the conservation of the PDF with time when the fibers are changing orientation. If we assume the fibers move with the bulk motion of the fluid then the PDF is a convected quantity and may be written as (A.18):

$$\frac{D\psi}{Dt} = -\nabla\left(\dot{\theta}\psi\right) - \frac{1}{\sin\theta}\nabla\left(\dot{\phi}\psi\right) = -\nabla\cdot\left(\psi\dot{\bar{p}}\right) \tag{A.18}$$

While the PDF gives a complete and unambiguous description of orientation, it can be cumbersome to calculate for large groups of fibers.

Taking the dyadic product of the single fiber orientation vector, $\bar{p}$, multiplying by the PDF and integrating of all possible orientation directions generates a set of orientation tensors. Since the PDF is periodic and even, the odd ordered integrals are zero. The second and forth order tensors are:

$$\bar{\bar{A}} = \oint p_i p_j \psi(\bar{p})\, d\bar{p} \tag{A.19}$$

$$\bar{\bar{A}}_4 = \oint p_i p_j p_k p_l \psi(\bar{p})\, d\bar{p} \tag{A.20}$$

Using the method above higher order orientation tensors are possible and can be encountered in certain stress models for fiber suspensions and orientation models [13, 14]. Further information about the orientation tensors can be found in Advani and Tucker [2].

A.3.2 Fiber Orientation Models

The finite difference MATLAB package contains a number of fiber orientation models than can be found in literature. This section outlines the models available and the differences involved with each model.

A.3.2.1 *Folgar-Tucker Model*

The Folgar-Tucker model is the earliest form of a model derived by Folgar and Tucker [15] using the orientation tensors described in § A.3.1. In (A.21), $\dot{\overline{\overline{A}}}$ is the material derivative $\left(\partial\overline{\overline{A}}/\partial t + \overline{v}\cdot\nabla\overline{\overline{A}}\right)$, $\overline{\overline{A}}$ is the 2nd order orientation tensor, $\overline{\overline{\mathbb{A}}}_4$ is the 4th order orientation tensor, $\overline{\overline{W}}$ is the vorticity, $\overline{\overline{D}}$ is the rate of deformation and $\overline{\overline{I}}$ is the identity tensor.

$$\dot{\overline{\overline{A}}} = \overline{\overline{W}}\cdot\overline{\overline{A}} - \overline{\overline{A}}\cdot\overline{\overline{W}} + \xi\left(\overline{\overline{D}}\cdot\overline{\overline{A}} + \overline{\overline{A}}\cdot\overline{\overline{D}} - 2\overline{\overline{\mathbb{A}}}_4 : \overline{\overline{D}}\right) + 2\dot{\gamma}C_I\left(\overline{\overline{I}} - 3\overline{\overline{A}}\right) \tag{A.21}$$

The variable ξ is a shape factor with the value given in (A.22):

$$\xi = \frac{a_r^2 + 1}{a_r^2 - 1} \tag{A.22}$$

The variable C_I is a phenomenological constant that accounts for fiber-fiber interaction in the concentrated suspension and is typically determined by fitting experimental fiber orientation or through transient stress fitting [6, 10]. Common values of the parameter can range from 0.003 – 0.016 [16]. Efforts have been made to fit the value through other means but the judgment of the experimenter must be used here [10, 17, 18].

A.3.2.2 *Modified Folgar-Tucker Model*

Stress growth experiments in the startup of simple shear have shown that the orientation of rigid fibers in concentrated suspension evolves more slowly than predicted by the Folgar-Tucker model [5, 19]. To more accurately reflect the observed transient fiber orientation a "slip" parameter was suggested by Sepehr et al. [20, 21]. The strain reduction factor (SRF) model proved more accurate in comparison to experimental data but is not objective.

$$\dot{\overline{\overline{A}}} = \kappa\left[\overline{\overline{W}}\cdot\overline{\overline{A}} - \overline{\overline{A}}\cdot\overline{\overline{W}} + \xi\left(\overline{\overline{D}}\cdot\overline{\overline{A}} + \overline{\overline{A}}\cdot\overline{\overline{D}} - 2\overline{\overline{\mathbb{A}}}_4 : \overline{\overline{D}}\right) + 2\dot{\gamma}C_I\left(\overline{\overline{I}} - 3\overline{\overline{A}}\right)\right] \tag{A.23}$$

The equations seen in (A.22) and (A.23) are identical with the exception of adding the "strain reduction parameter", κ, to (A.23). This value is usually fit using experimental orientation data or through transient stress fitting.

A.3.2.3 *Reduced Strain Closure Model*

In an effort to make (A.23) into an objective form, Wang et al. [22] proposed the "reduced strain closure" model whereby the strain reduction factor was rearranged so that it acted only on the objective tensors of the equation. The resulting model is (A.24):

$$\begin{aligned}\dot{\overline{\overline{A}}} = {} & \overline{\overline{W}}\cdot\overline{\overline{A}} - \overline{\overline{A}}\cdot\overline{\overline{W}} + \xi\left\{\overline{\overline{D}}\cdot\overline{\overline{A}} + \overline{\overline{A}}\cdot\overline{\overline{D}} - 2\left[\overline{\overline{\mathbb{A}}}_4 + (1-\kappa)\left(\overline{\overline{\mathbb{L}}}_4 - \overline{\overline{\mathbb{M}}}_4 : \overline{\overline{\mathbb{A}}}_4\right)\right] : \overline{\overline{D}}\right\}\ldots \\ & + 2\kappa\dot{\gamma}C_I\left(\overline{\overline{I}} - 3\overline{\overline{A}}\right)\end{aligned} \tag{A.24}$$

The new terms in (A.24) $\left(\overline{\overline{\mathbb{L}}}_4, \overline{\overline{\mathbb{M}}}_4\right)$ are the eigenvalues, λ_i, and eigenvectors, $\overline{e}_i$, of the 2nd order orientation tensor.

$$\overline{\overline{\mathbb{L}}}_4 = \sum_{i=1}^{3} \lambda_i \overline{e}_i \overline{e}_i \overline{e}_i \overline{e}_i \tag{A.25}$$

$$\overline{\overline{\mathbb{M}}}_4 = \sum_{i=1}^{3} \overline{e}_i \overline{e}_i \overline{e}_i \overline{e}_i \tag{A.26}$$

A.3.2.4 *Anisotropic Rotary Diffusion Reduced Strain Closure Model*

The previous models utilized C_I to account for fiber-fiber interaction in concentrated fiber suspensions. This term assumes that the material is isotropic but this assumption is rarely valid in glass fiber systems. Phelps and Tucker developed a model where by the fiber interaction coefficient is a 2nd order tensor to more accurately reflect the anisotropic nature of fiber composite materials. The ARD-RSC model is given in (A.27):

$$\begin{aligned}\dot{\overline{\overline{A}}} = {} & \overline{\overline{W}} \cdot \overline{\overline{A}} - \overline{\overline{A}} \cdot \overline{\overline{W}} + \xi \left\{ \overline{\overline{D}} \cdot \overline{\overline{A}} + \overline{\overline{A}} \cdot \overline{\overline{D}} - 2 \left[\overline{\overline{\mathbb{A}}}_4 + (1-\kappa) \left(\overline{\overline{\mathbb{L}}}_4 - \overline{\overline{\mathbb{M}}}_4 : \overline{\overline{\mathbb{A}}}_4 \right) \right] : \overline{\overline{D}} \right\} \ldots \\ & + \dot{\gamma} \left\{ 2 \left[\overline{\overline{C}} - (1-\kappa) \overline{\overline{\mathbb{M}}}_4 : \overline{\overline{C}} \right] - 2\kappa \left(\mathrm{tr} \overline{\overline{C}} \right) \overline{\overline{A}} - 5 \left(\overline{\overline{C}} \cdot \overline{\overline{A}} + \overline{\overline{A}} \cdot \overline{\overline{C}} \right) \ldots \right. \\ & \left. + 10 \left[\mathbb{A}_4 + (1-\kappa) \left(\overline{\overline{\mathbb{L}}}_4 - \overline{\overline{\mathbb{M}}}_4 : \overline{\overline{\mathbb{A}}}_4 \right) \right] : \overline{\overline{C}} \right\}\end{aligned} \tag{A.27}$$

Where the fiber interaction tensor, $\overline{\overline{C}}$, is given in (A.28):

$$C = b_1 \overline{\overline{I}} + b_2 \overline{\overline{A}} + b_3 \overline{\overline{A}}^2 + \frac{b_4}{\dot{\gamma}} \overline{\overline{D}} + \frac{b_5}{\dot{\gamma}^2} \overline{\overline{D}}^2 \tag{A.28}$$

The values of $b_1 - b_5$ are fit through similar methods as C_I.

A.3.2.5 *Semi-Flexible Fiber Model (Concentrated Bead-Rod Model)*

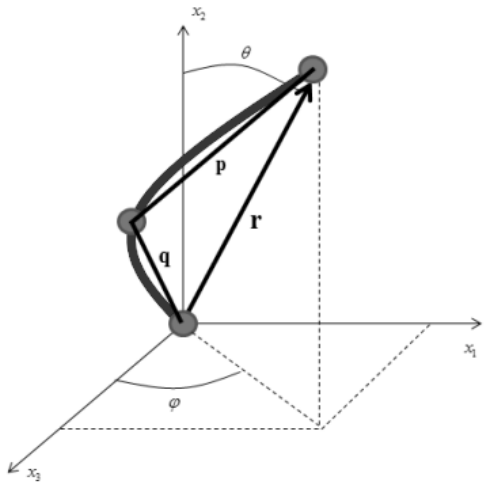

Figure A.3: Vector definition of flexible fiber. The fiber is characterized by the vectors, $\overline{p}$ and $\overline{q}$, as well as the azimuthal and zenith angles, ϕ and θ respectively. The two vector are of

equal length, l_b, and there is an internal resistivity to bending, k. The end-to-end vector, $\overline{r}$, is also defined to determine a fiber's "average" direction.

The method of orientation tensors presented by Advani and Tucker [2] was adapted by Strautins and Latz [23] shown in (A.29) to (A.31) where $\overline{p}$ and $\overline{q}$ represent the two vectors of the bent fiber (shown in) and $\psi(\overline{p},\overline{q},t)$ is the probability distribution function for an individual fiber.

$$\overline{\overline{A}} = \oint \overline{p}\overline{p}\psi(\overline{p},\overline{q},t)\,d\overline{p}\,d\overline{q} \tag{A.29}$$

$$\overline{\overline{B}} = \oint \overline{p}\overline{q}\psi(\overline{p},\overline{q},t)\,d\overline{p}\,d\overline{q} \tag{A.30}$$

$$\overline{C} = \oint \overline{p}p\psi(\overline{p},\overline{q},t)\,d\overline{p}\,d\overline{q} \tag{A.31}$$

Ortman et al. [24] adapted the orientation equations given by Strautins and Latz [23] to concentrated suspensions by including the isotropic rotary diffusion term and slip parameter from previous work [9, 20]. The adaptation utilizes the orientation tensors in (A.29) - (A.31) and is given by (A.32) - (A.35) where l_b is the length of one rod of the two rod system.

$$\begin{aligned}\dot{\overline{\overline{A}}} = \kappa\Big[&\overline{\overline{W}}\cdot\overline{\overline{A}} - \overline{\overline{A}}\cdot\overline{\overline{W}} + \xi\left(\overline{\overline{D}}\cdot A + \overline{\overline{A}}\cdot\overline{\overline{D}} - 2\overline{\overline{A}}_4 : \overline{\overline{D}}\right) - 6C_I\dot{\gamma}\left(\overline{\overline{A}} - \tfrac{1}{3}\overline{\overline{I}}\right)\ldots \\ &+\frac{l_b}{2}\Big[\overline{C}\overline{m} + \overline{m}\overline{C} - 2\left(\overline{m}\cdot\overline{C}\right)\overline{\overline{A}}\Big] + 2k\left(\overline{\overline{B}} - \overline{\overline{A}}tr\left(\overline{\overline{B}}\right)\right)\Big]\end{aligned} \tag{A.32}$$

$$\begin{aligned}\dot{\overline{\overline{B}}} = \kappa\Big[&\overline{\overline{W}}\cdot\overline{\overline{B}} - \overline{\overline{B}}\cdot\overline{\overline{W}} + \xi\left(\overline{\overline{D}}\cdot\overline{\overline{B}} + \overline{\overline{B}}\cdot\overline{\overline{D}} - 2\left(\overline{\overline{D}} : \overline{\overline{A}}\right)\overline{\overline{B}}\right) - 4C_I\dot{\gamma}\overline{\overline{B}}\ldots \\ &+\frac{l_b}{2}\Big[\overline{C}\overline{m} + \overline{m}\overline{C} - 2\left(\overline{m}\cdot\overline{C}\right)\overline{\overline{A}}\Big] + 2k\Big[\overline{\overline{A}} - \overline{\overline{B}}\ \mathrm{tr}\left(\overline{\overline{B}}\right)\Big]\Big]\end{aligned} \tag{A.33}$$

$$\dot{\overline{C}} = \kappa\left[\nabla\overline{v}^T\cdot\overline{C} - \left(\overline{\overline{A}} : \nabla\overline{v}^T\right)\overline{C} + \frac{l_b}{2}\Big[\overline{m} - \overline{C}\left(\overline{m}\cdot\overline{C}\right)\Big] - k\overline{C}\Big[1 - \mathrm{tr}\left(\overline{\overline{B}}\right)\Big] - 2C_I\dot{\gamma}\overline{C}\right] \tag{A.34}$$

$$\overline{m} = \sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3}\frac{\partial^2 v_i}{\partial x_j\,\partial x_k}A_{jk}\overline{\delta}_i \tag{A.35}$$

The internal resistivity to bending, k, is a coefficient that attempts to account for the physical bending that may occur in the system of fibers. As the value of k increases, the semi-flexible model behaves more like the rigid fiber model and in the limit of $k \to \infty$ the semi-flexible model parallels the rigid fiber model. Conversely, as the value of k decreases the fiber becomes much more flexible and in the limit of $k \to 0$ the fiber is completely flexible. In the equations presented above, the flexibility of the fiber is initiated in the hydrodynamic contributions given by (A.35) and is due to the bending that may occur because of gradients in the flow field.

A.3.3 Numerical Method

Within this section, a number of the numerical techniques used in the finite difference calculations will be discussed.

A.3.3.1 *Choice of Integration Routine*

The integration technique utilized for the simulations is the 'ode45' or 'ode15s' built-in routine in the MATLAB environment. The ode45 routine is the Range-Kutta 4,5 technique that can be found in any number of numerical mathematics books. The 'ode15s' routine is used when 'ode45' runs slow or has poor convergence and has a number of advantages over the Range-Kutta method including:

- Allows for specification of both relative and absolute tolerance.
- Allows for vectorization of ODE system.
- Automatic time step adjustment during integration.
- Calculation of Jacobian matrix when necessary.
- Calculation of mass matrix.
- Allows specification of NDF or BDF type differentiation.
- Allows for specification of max order differentiation for numerically unstable problems.
- Solves differential equations of the form: $M(t,y)y' = f(t,y)$

A.3.3.2 *Fiber Orientation Prediction Boundary Conditions and Derivatives*

Two derivatives were estimated though the use of finite differences. In all of the following examples the super scripts and subscripts may be changed to calculate different derivatives as long as the pattern is followed. It is important to note that the grid spacing for the finite difference calculations was equal through out with a size of 0.045 mm x 0.045 mm x 0.045 mm.

Discretization of $\underline{m}$ vector was carried out through the use of central, forward and backward difference methods. In the hexagonal mesh, i is the index of a particular x coordinate (flow direction), j is the index of a particular y coordinate (vertical direction) and k is the index of a particular z coordinate (transverse to flow direction). Furthermore, v is any component of the velocity vector, Δx is the distance between nodes in the primary flow direction and Δy is the distance between nodes in the vertical direction.

Interior Nodes – Central Difference

$$\frac{\partial^2 v}{\partial x \partial y}\left(x^i, y^j, z^k\right) = \frac{v^{i+1,j+1,k} - v^{i-1,j+1,k} - v^{i+1,j-1,k} + v^{i-1,j-1,k}}{4\left(\Delta x \Delta y\right)}$$

$$\frac{\partial^2 v}{\partial x^2}\left(x^i, y^j, z^k\right) = \frac{v^{i+1,j,k} - 2v^{i,j,k} + v^{i-1,j,k}}{\left(\Delta x\right)^2}$$

Boundary Nodes – Forward or Backward Difference

$$\frac{\partial^2 v}{\partial x \partial y}\left(x^i, y^j, z^k\right) = \frac{v^{i,j,k} - v^{i-1,j,k} - v^{i,j-1,k} + v^{i-1,j-1,k}}{\Delta x \Delta y}$$

$$\frac{\partial^2 v}{\partial x \partial y}\left(x^i, y^j, z^k\right) = \frac{v^{i+1,j+1,k} - v^{i,j+1,k} - v^{i+1,j,k} + v^{i,j,k}}{\Delta x \Delta y}$$

$$\frac{\partial^2 v}{\partial x^2}\left(x^i, y^j, z^k\right) = \frac{v^{i,j,k} - 2v^{i-1,j,k} + v^{i-2,j,k}}{\left(\Delta x\right)^2}$$

$$\frac{\partial^2 v}{\partial x^2}\left(x^i, y^j, z^k\right) = \frac{v^{i+2,j,k} - 2v^{i+1,j,k} + v^{i,j,k}}{\left(\Delta x\right)^2}$$

Discretization of the $\underline{v} \cdot \nabla \underline{\underline{A}}$ is carried out through the use of backward and forward (shown in order below) difference methods with examples for the A_{12} component of orientation and the v component of velocity. In the hexagonal mesh, i is the index of a particular x coordinate (flow direction), j is the index of a particular y coordinate (vertical direction) and k is the index of a particular z coordinate (transverse to flow direction). Furthermore, v is any component of the velocity vector and Δx is the distance between nodes transverse to the primary flow direction.

$$v\frac{\partial A_{12}}{\partial x}\left(x^i, x^j, x^k\right) = v\left(\frac{A_{12}^{i-2,j,k} - 4A_{12}^{i-1,j,k} + 3A_{12}^{i,j,k}}{2\Delta x}\right)$$

$$v\frac{\partial A_{12}}{\partial x}\left(x^i, x^j, x^k\right) = v\left(\frac{A_{12}^{i,j,k} - 4A_{12}^{i+1,j,k} + 3A_{12}^{i+2,j,k}}{2\Delta x}\right)$$

Boundary Conditions

- Boundary condition at the wall is no slip. The temperature of the wall is is prescribed as $T_{\text{WALL}} = 363K$
- Boundary condition at free surface is given in terms of $\underline{\underline{\sigma}} = -P\underline{\underline{I}} + \underline{\underline{\tau}}$ where $\underline{\underline{\sigma}}$ is the total stress tensor, P is the isotropic pressure, $\underline{\underline{I}}$ is the unit tensor and $\underline{\underline{\tau}}$ is the extra stress tensor:
    - Use a zero traction vector at the front, i.e $\underline{n} \bullet \underline{\underline{\sigma}} \bullet \underline{n} = 0$ and $\underline{\underline{\sigma}} \bullet \underline{n} - \left(\underline{n} \bullet \underline{\underline{\sigma}} \bullet \underline{n}\right)\underline{n} = \underline{0}$.
    - Applied where the fluid fraction is zero.
    - The temperature equation at the front is solved with a prescribed temperature of 300 K.
- The volumetric flow rate was given at the inlet to the domain and a fully developed velocity profile was assumed. Furthermore, an initial temperature was provided at the inlet to the domain.

The initial temperature is given as: $T_{\text{IN}} = 463K$

The volumetic flow rate at the inlet is given by: $Q_{\text{IN}} = \dfrac{\text{Total Volume of Mold}}{\text{Desired Fill Time of Mold}}$

A.3.4 Fiber Orientation Simulation Output

When the selected fiber orientation prediction script is run the data is output at each time step to a specified directory. A person simulating should be comfortable with multi-dimensional (4-5 dimensional) arrays inside the MATLAB environment. To date no explicit script has been written to help in visualizing results from the simulations since different studies will require different comparisons. An example of visualizing orientation data is given in Figure A.4.

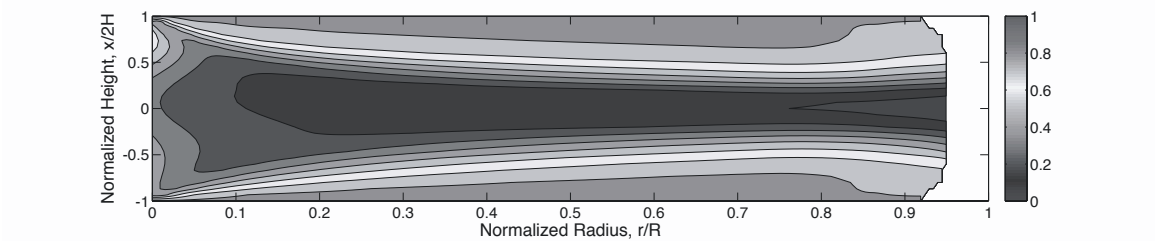

Figure A.4: Example of post-processing fiber orientation data to visualize cross-section of center-gated disk. Contours represent value of A_{11} orientation component.

A.3.5 Automated Transient Stress Fitting

The method used to fit transient stress overshoots in simple shear flow has the complexity of producing multiple solutions. To combat this, an unbiased way of fitting was attempted through the use on non-linear least squares fitted (NLLSQ) analysis using the 'lsqnonlin' function within the MATLAB environment. There are two stress models that can be used during the fitting of the transient stress over shoots provided by Dinh and Armstrong [25] and Ortman et al. [24] shown in (A.36) and respectively (A.37).

$$\overline{\overline{\tau}} = 2\eta_m \left(\overline{\overline{D}} + c_1 \phi \overline{\overline{D}} + c_2 \phi \overline{\overline{D}} : \overline{\overline{A}}_4 \right) \tag{A.36}$$

$$\overline{\overline{\tau}} = 2\eta_m \left(\overline{\overline{D}} + f_1 \phi \overline{\overline{D}} + f_2 \overline{\overline{A}}_4 : \overline{\overline{D}} \right) + c_3 \eta_m k \frac{3\phi a_r}{2} \frac{\text{tr}\left(\overline{r}\right)}{2l_b^2} \left(\overline{\overline{A}} - \overline{\overline{R}} \right) \tag{A.37}$$

Where,

$$\overline{\overline{R}} = \frac{\langle \overline{rr} \rangle}{\text{tr}\left(\overline{r}\right)} = \frac{\overline{\overline{A}} - \overline{\overline{B}}}{1 - \text{tr}\left(\overline{\overline{B}}\right)} \tag{A.38}$$

The results of such a fitting procedure are shown below in Figure A.5.

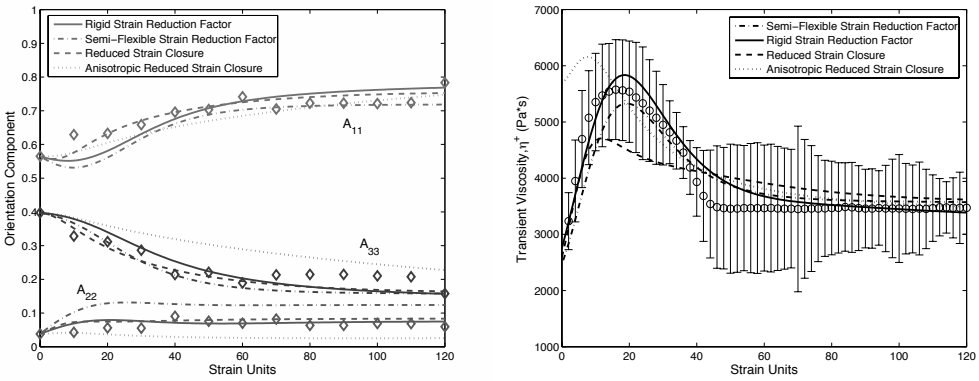

Figure A.5: NLLSQ fitting of experimental orientation data (left) and transient viscosity (right) for various fiber orientation equations.

A.3.6 Batch Processing

The solution of the fiber orientation equations, especially when simulating the full end-gated plaque, can take upwards of 24-48 hours. This is a long time to keep the MATLAB interface window in a "busy" state. To run the simulation scripts and keep the MATLAB interface open to other work use the batch scheduling ability of MATLAB to run the simulations on other cores. The batch processing ability can be accessed by entering the following:

sched = findResource('scheduler','type','local');

job = batch(sched,'D:\...\myscript.m');

A.4 Complete List of Files

A.4.1 Interface Scripts

- cgd_ft.m (Center-gated disk simulation, FT model)
- cgd_br.m (Center-gated disk simulation, BR model)
- cgd_rsc.m (Center-gated disk simulation, RSC model)
- cgd_ard.m (Center-gated disk simulation, ARD-RSC model)
- egp_ft.m (End-gated plaque simulation, FT model)
- egp_br.m (End-gated plaque simulation, BR model)
- egp_rsc.m (End-gated plaque simulation, RSC model)
- egp_ard.m (End-gated plaque simulation, ARD-RSC model)

A.4.2 Automated Fitting Scripts

- stress_orient_fit.m (NLLSQ Fitting of Sliding Plate Data, Rigid Models)
- stress_orient_fit_br.m (NLLSQ Fitting of Sliding Plate Data, Semi-flexible Models)
- param_fit.m (NLLSQ Fitting of Experimental Orientation Data, Rigid Models)
- param_fit_br.m (NLLSQ Fitting of Experimental Orientation Data, Semi-flexible Models)

A.4.3 Model Files

- FT_quad.m (Modified FT Model, Quadratic Closure)
- FT_IBOF.m (Modified FT Model, IBOF Closure)

- FT_quadc.m (Modified FT Model, Quadratic Closure, w/ Convection)
- FT_IBOFc.m (Modified FT Model, IBOF Closure, w/ Convection)
- BR_quad.m (Modified BR Model, Quadratic Closure)
- BR_IBOF.m (Modified BR Model, IBOF Closure)
- BR_quadc.m (Modified BR Model, Quadratic Closure, w/ Convection)
- BR_IBOFc.m (Modified BR Model, IBOF Closure, w/ Convection)
- RSC_quad.m (RSC Model, Quadratic Closure)
- RSC_IBOF.m (RSC Model, IBOF Closure)
- RSC_quadc.m (RSC Model, Quadratic Closure, w/ Convection)
- RSC_IBOFc.m (RSC Model, IBOF Closure, w/ Convection)
- ARD_RSC_quad.m (ARD-RSC Model, Quadratic Closure)
- ARD_RSC_IBOF.m (ARD-RSC Model, IBOF Closure)
- ARD_RSC_quadc.m (ARD-RSC Model, Quadratic Closure, w/ Convection)
- ARD_RSC_IBOFc.m (ARD-RSC Model, IBOF Closure, w/ Convection)

A.4.4 Ancillary Files

- convective_v.m (calculation of convection for given A_{ij}, B_{ij}, vectorized)
- mvector.m (calculation of $\bar{m}$ for Bead-Rod Model)
- Explicit_IBOF_v.m (IBOF Closure Approximation $\bar{\bar{\mathbb{A}}}_4 : \bar{\bar{D}}$, vectorized)
- Explicit_Quad_v.m (Quadratic Closure Approximation $\bar{\bar{\mathbb{A}}}_4 : \bar{\bar{D}}$, vectorized)
- LddC.m $\left(\bar{\bar{\mathbb{L}}}_4 : \bar{\bar{C}}\right)$
- LddD.m $\left(\bar{\bar{\mathbb{L}}}_4 : \bar{\bar{D}}\right)$
- MddC.m $\left(\bar{\bar{\mathbb{M}}}_4 : \bar{\bar{C}}\right)$
- MddAddC.m $\left(\bar{\bar{\mathbb{M}}}_4 : \bar{\bar{\mathbb{A}}}_4 : \bar{\bar{C}}, \text{ Quadratic Closure Approximation}\right)$
- MddAddC_IBOF.m $\left(\bar{\bar{\mathbb{M}}}_4 : \bar{\bar{\mathbb{A}}}_4 : \bar{\bar{C}}, \text{ IBOF Closure Approximation}\right)$
- MddAddD.m $\left(\bar{\bar{\mathbb{M}}}_4 : \bar{\bar{\mathbb{A}}}_4 : \bar{\bar{D}}, \text{ Quadratic Closure Approximation}\right)$
- MddAddD_IBOF.m $\left(\bar{\bar{\mathbb{M}}}_4 : \bar{\bar{\mathbb{A}}}_4 : \bar{\bar{D}}, \text{ IBOF Closure Approximation}\right)$
- timestepconverter.m (HTML to Text Conversion)

A.5 References

[1] R. B. Bird, "Citation Classic - Dynamics of Polymeric Liquids, Vol 1, Fluid-Mechanics, Vol 2, Kinetic-Theory," *Current Contents/Engineering Technology & Applied Sciences,* pp. 18-18, Aug 22 1988.

[2] S. G. Advani and C. L. Tucker, "The Use of Tensors to Describe and Predict Fiber Orientation in Short Fiber Composites," *Journal of Rheology,* vol. 31, pp. 751-784, Nov 1987.

[3] B. E. VerWeyst and C. L. Tucker, "Fiber suspensions in complex geometries: Flow/orientation coupling," *Canadian Journal of Chemical Engineering,* vol. 80, pp. 1093-1106, Dec 2002.

[4] D. H. Chung and T. H. Kwon, "Numerical studies of fiber suspensions in an axisymmetric radial diverging flow: the effects of modeling and numerical assumptions," *Journal of Non-Newtonian Fluid Mechanics,* vol. 107, pp. 67-96, Dec 6 2002.

[5] D. G. Baird, A. P. R. Eberle, G. M. Velez-Garcia, and P. Wapperom, "Fiber orientation kinetics of a concentrated short glass fiber suspension in startup of simple shear flow," *Journal of Non-Newtonian Fluid Mechanics,* vol. 165, pp. 110-119, Feb 2010.

[6] D. G. Baird, A. P. R. Eberle, P. Wapperom, and G. M. Velez-Garcia, "Using transient shear rheology to determine material parameters in fiber suspension theory," *Journal of Rheology,* vol. 53, pp. 685-705, May-Jun 2009.

[7] K. C. Ortman, P. Wapperom, A. Whittington, and D. G. Baird, "Using startup of steady shear flow in a sliding plate rheometer to determine material parameters for the purpose of predicting long fiber orientation," *Journal of Rheology,* vol. 56, pp. 955-981, 2012.

[8] C. W. Hirt and B. D. Nichols, "Volume of Fluid (Vof) Method for the Dynamics of Free Boundaries," *Journal of Computational Physics,* vol. 39, pp. 201-225, 1981.

[9] R. S. Bay and C. L. Tucker, "Fiber Orientation in Simple Injection Moldings .1. Theory and Numerical-Methods," *Polymer Composites,* vol. 13, pp. 317-331, Aug 1992.

[10] R. S. Bay and C. L. Tucker, "Fiber Orientation in Simple Injection Moldings .2. Experimental Results," *Polymer Composites,* vol. 13, pp. 332-341, Aug 1992.

[11] J. L. White and J. E. Spruiell, "Specification of Biaxial Orientation in Amorphous and Crystalline Polymers," *Polymer Engineering and Science,* vol. 21, pp. 859-868, 1981.

[12] J. L. White and J. E. Spruiell, "The Specification of Orientation and Its Development in Polymer Processing," *Polymer Engineering and Science,* vol. 23, pp. 247-256, 1983.

[13] M. P. Petrich, D. L. Koch, and C. Cohen, "An experimental determination of the stress-microstructure relationship in semi-concentrated fiber suspensions," *Journal of Non-Newtonian Fluid Mechanics,* vol. 95, pp. 101-133, Dec 25 2000.

[14] D. L. Koch, "A Model for Orientational Diffusion in Fiber Suspensions," *Physics of Fluids,* vol. 7, pp. 2086-2088, Aug 1995.

[15] F. Folgar and C. L. Tucker, "Orientation Behavior of Rigid Fibers in Concentrated Suspensions," *Journal of Rheology,* vol. 26, pp. 604-604, 1982.

[16] R. G. Larson, *The structure and rheology of complex fluids*. New York: Oxford University Press, 1999.

[17] C. L. Tucker and J. H. Phelps, "An anisotropic rotary diffusion model for fiber orientation in short- and long-fiber thermoplastics," *Journal of Non-Newtonian Fluid Mechanics,* vol. 156, pp. 165-176, Feb 2009.

[18] B. E. VerWeyst, C. L. Tucker, P. H. Foss, and J. F. O'Gara, "Fiber orientation in 3-D injection molded features - Prediction and experiment," *International Polymer Processing,* vol. 14, pp. 409-420, Dec 1999.

[19] M. Sepehr, P. J. Carreau, M. Moan, and G. Ausias, "Rheological properties of short fiber model suspensions," *Journal of Rheology,* vol. 48, pp. 1023-1048, Sep-Oct 2004.

[20] M. Sepehr, G. Ausias, and P. J. Carreau, "Rheological properties of short fiber filled polypropylene in transient shear flow," *Journal of Non-Newtonian Fluid Mechanics,* vol. 123, pp. 19-32, Oct 15 2004.

[21] M. Sepehr, P. J. Carreau, M. Grmela, G. Ausias, and P. G. Lafleur, "Comparison of rheological properties of fiber suspensions with model predictions," *Journal of Polymer Engineering,* vol. 24, pp. 579-610, Nov-Dec 2004.

[22] J. Wang, J. F. O'Gara, and C. L. Tucker, "An objective model for slow orientation kinetics in concentrated fiber suspensions: Theory and rheological evidence," *Journal of Rheology,* vol. 52, pp. 1179-1200, Sep-Oct 2008.

[23] U. Strautins and A. Latz, "Flow-driven orientation dynamics of semiflexible fiber systems," *Rheologica Acta,* vol. 46, pp. 1057-1064, Oct 2007.

[24] K. Ortman, D. Baird, P. Wapperom, and A. Whittington, "Using startup of steady shear flow in a sliding plate rheometer to determine material parameters for the purpose of predicting long fiber orientation," *Journal of Rheology,* vol. 56, pp. 955-981, Jul 2012.

[25] S. M. Dinh and R. C. Armstrong, "A Rheological Equation of State for Semiconcentrated Fiber Suspensions," *Journal of Rheology,* vol. 28, pp. 207-227, 1984.

Appendix: Dimensional Analysis and Comments on Numerical Routine

B Appendix: Comments on Numerical Routine

B.1 Dimensional Analysis for Complex Geometry Simulations

In simulations involving significant computational resources, dimensional analysis is a simple way of speeding up computation time by removing certain terms in the general set of equations that are orders of magnitude smaller than others. In the following section an analysis of terms for both the heat and motion equations will be performed for general cases of non-Newtonian flow in complex geometries.

B.1.1 Dimensional Analysis for Equation of Motion

The most general form of the equation of motion is given in Eq. (B.1) where the left-hand side represents the inertial terms and the right-hand side represents the viscous terms where ρ is the density, $\bar{v}$ is the velocity, P is the pressure, and $\bar{\bar{\tau}}$ is deviatoric stress tensor.

$$\underbrace{\overbrace{\rho\frac{\partial \bar{v}}{\partial t}}^{\text{Unsteady Acceleration}} + \overbrace{\rho\left(\bar{v}\cdot\nabla\bar{v}\right)}^{\text{Convective Acceleration}}}_{\text{Intertial Contribution}} = \underbrace{\overbrace{-\nabla P}^{\text{Pressure Gradient}} + \overbrace{\nabla\cdot\bar{\bar{\tau}}}^{\text{Viscosity}}}_{\text{Divergence of Stress}} \tag{B.1}$$

By assuming constant density and Stokes flow $\left(\mathrm{Re}<1\right)$, Eq. (B.1) can be rewritten for laminar flow where the inertial contributions become orders of magnitude smaller than the terms on the right-hand side and thus Eq. (B.2) is produced.

$$0 = -\nabla P + \nabla\cdot\bar{\bar{\tau}} \tag{B.2}$$

While the reduction of terms in the momentum equation is not significant for most modern workstations or clusters, when the motion equations are coupled with other conservations equations (i.e. stress, heat, orientation) the Stokes flow case can reduce computation time and convergence issues.

B.1.2 Dimensional Analysis for Equation of Heat

Dimensional analysis for the heat equation can provide useful insight into which terms of the equation can be kept and which can be dropped. The most general form of the energy equation is shown in Eq. (B.3) where C_p is the heat capacity, k is the thermal conductivity, T is the temperature, $\bar{\bar{\tau}}$ is deviatoric stress tensor, and $\bar{\bar{\dot{\gamma}}}$ is the rate-of-deformation tensor or $\bar{\bar{\dot{\gamma}}} = \nabla\bar{v} + \left(\nabla\bar{v}\right)^T$.

$$\underbrace{\rho C_p\frac{DT}{Dt}}_{\text{Convection}} = \underbrace{k\nabla^2 T}_{\text{Conduction}} + \underbrace{\bar{\bar{\tau}}:\bar{\bar{\dot{\gamma}}}}_{\text{Dissipation}} \tag{B.3}$$

The heat equation can be rewritten in a dimensionless form by making the assumption that the viscosity of the fluid is dependent on both temperature and shear-rate or $\eta\left(T,\dot{\gamma}\right)$. Using dimensionless variables, described by Winter [1], the dimensionless form of Eq. (B.3) can be written yielding Eq. (B.4) where the GNF model for viscosity has been used for and η^0 is the

reference viscosity, ΔT^0 is the reference temperature difference, V is the reference velocity, and H and L are the characteristic height and length, respectively. Also the substitutions for the dimensionless variables are: $T^* = (T - T_0)/\Delta T^0$, $t^* = tL/V$, $\nabla^* = H\nabla$, $\eta^* = \eta/\eta^0$, and $\dot{\gamma}^* = \dot{\gamma} H/V$. These two dimensionless groups become important in determining if convection and viscous dissipation terms should be included in the system being solved.

$$\underbrace{\frac{\rho C_p V H}{k}}_{\text{Graetz Number}} \frac{H}{L}\frac{DT^*}{Dt^*} = \nabla^* T^* + \frac{1}{2}\underbrace{\frac{V^2\eta^0}{k\Delta T^0}}_{\text{Generation Number}} \eta^*\left(\dot{\gamma}^* : \dot{\gamma}^*\right) \tag{B.4}$$

Thermal convection can usually be neglected if the Graetz number is much less than one, or $\mathrm{Gr} << 1$. For many mold filling operations the Hele-Shaw and lubrication approximations are employed. Here the same reasoning for those approximations will be used, that the length of the cavity is orders of magnitude larger than the thickness, or $L/H >> 1$. At first it may seem appropriate to remove the convective term from the equation but mold filling velocities are usually high to ensure quick fill times. For the polymer systems of interest, the Graetz number is estimated at the mold entrance to be around 100 so that thermal convection cannot be neglected, at least not in the mold entry region. This theory is further reinforced by use of the Péclet number where estimates are on the order of 10^3-10^4 and is defined as the ratio of thermal advection to diffusion given by Eq. (B.5).

$$\mathrm{Pe} = \frac{LV}{k/\rho C_p} \tag{B.5}$$

Viscous dissipation is usually a concern when dealing with polymeric systems because of the non-linear nature of their viscosities. If the problem is defined as a molding operation, where the temperature of the mold is a known quantity, the reference temperature difference is given as $\Delta T_{\text{process}} = T_{\text{fluid}} - T_{\text{wall}}$ and the generation number is redefined as the Brinkman number, Br, and is defined by Eq. (B.6). The Brinkman number can also be formed by the product of the Prandtl number, Pr, and Eckert number, Ec.

$$\mathrm{Br} = \frac{V^2\eta^0}{k\Delta T_{\text{process}}} \tag{B.6}$$

The Brinkman number for molding operations can vary depending on neat matrix and any additive that may be present. For the fiber filled systems of interest, the Brinkman number is on the order of 10^2-10^3 and thus cannot be neglected.

To simplify the system of equations, the ratio of convection to generation may be compared on a case-by-case basis. The ratio of the generation (Brinkman) to the rate of convection (Graetz) is around one order of magnitude. This is not sufficiently large enough to neglect one term or the other so both will be kept when simulations are carried out [2].

B.2 Improvements to the FEM Solution Method

B.2.1 Domain Meshing

One of the most dominating characteristics in FEM computations, no matter how efficiently the code has been written, is how the domain of interest is meshed. As evidence of the requirement of efficient meshing, Figure B.1 shows the number of nodes and computational time as a function optimized meshes where an increasing mesh number represents a higher degree of optimization.

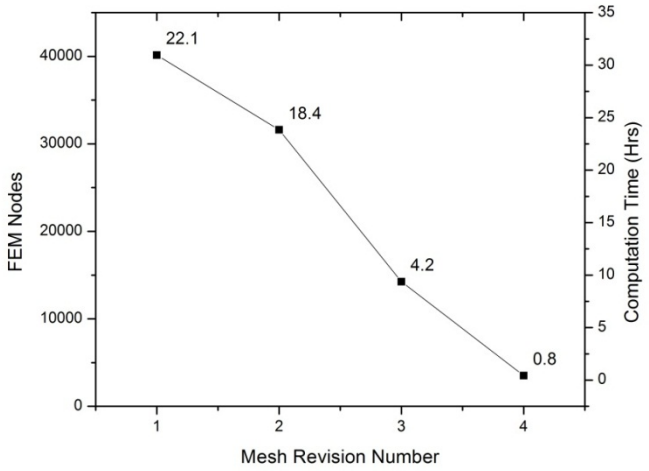

Figure B.1: Computational Time and FEM Nodes vs. Number of Refinements for center-gated disk. Revision 1: tetrahedral meshing only, automated meshing From ANSYS meshing utility. Revision 2: Quad/Tet meshing combination, manual meshing. Revision 3: hexagonal dominant meshing, automated meshing. Revision 4: Hexagonal meshing only, manual meshing.

When the ANSYS meshing utility is allowed to mesh the domain automatically the resulting mesh is cumbersome and takes hours of computation time for Newtonian and GNF fluids. The improvements to the generated meshes came through the use of different element shapes and sizes, as well as manually defining where coarse and fine meshing was required, depending on the location within the domain. For example, it is more prudent to mesh with bias towards the domain boundary because of the large gradients that occur near the wall as seen in Figure B.2.

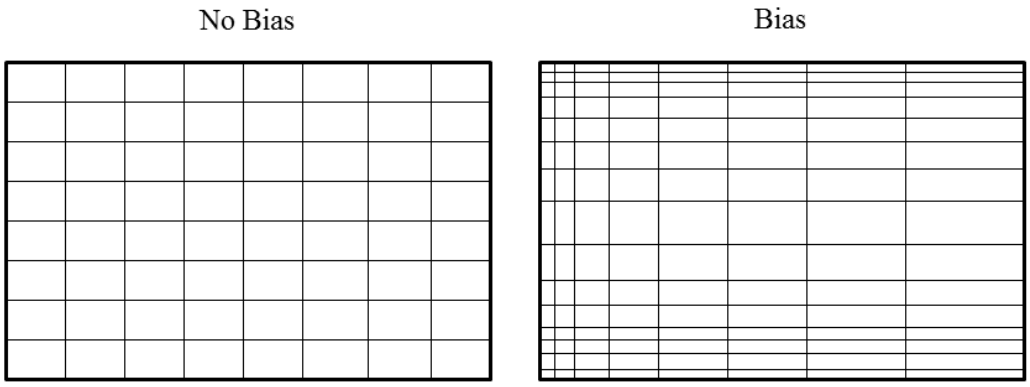

Figure B.2: Finite Element Meshes over a Generic Domain. (Left) Mesh exhibiting no bias. (Right) Mesh exhibiting bias in one horizontal direction (from left to right) and two biases in the vertical direction (near the walls).

The two test geometries which have been heavily simulated throughout this work are the center-gated disk and end-gated plaque. The figures included show the optimized meshes for the center-gated sprue, end-gated sprue, center-gated mold, and end-gated mold, respectfully.

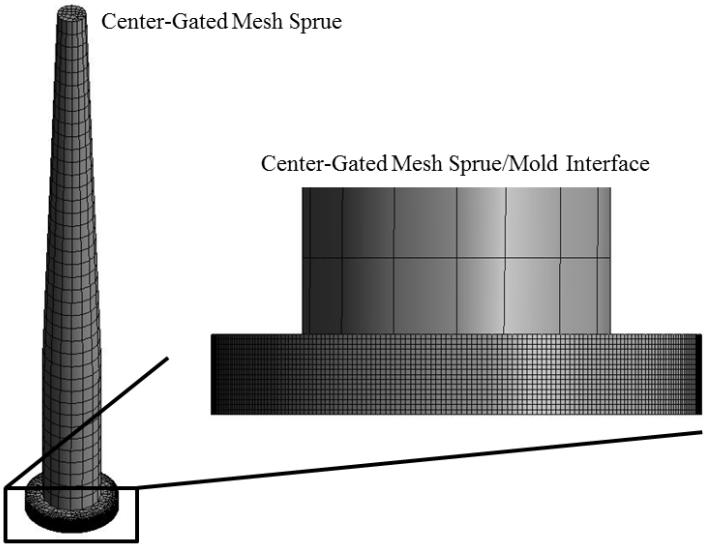

Figure B.3: FEM meshes for CGD sprue (left) and CGD sprue/mold interface (right).

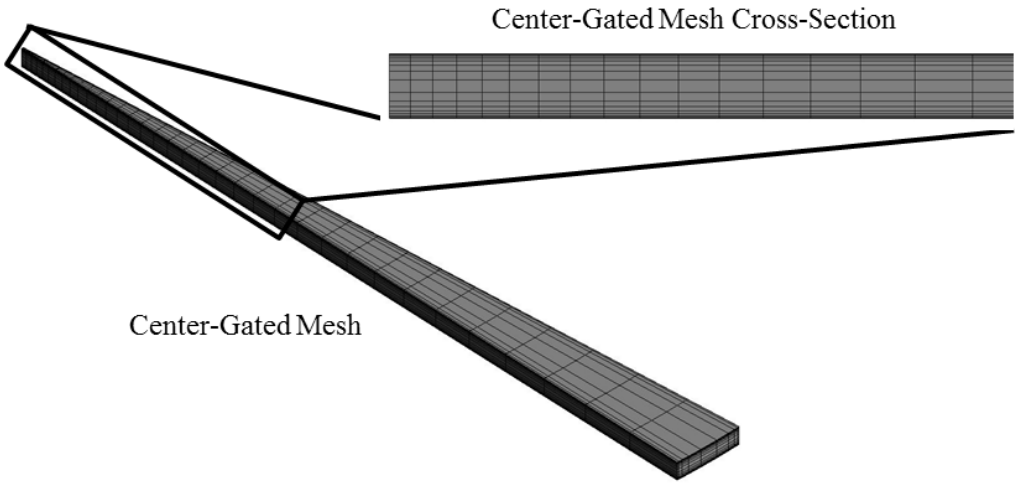

Figure B.4: Axisymmetric center-gated mold FEM mesh. (Left) The mold filling simulation begins from the left edge of the mold and progresses through to the right edge. (Right) Close-up of the mold cavity thickness noting the bias shown towards the walls of the mold.

B.2.2 **Meshing Analysis**

For finite difference and finite element analyses a balance must be struck between the number of elements and computational accuracy. Generally, a higher number of elements in a domain yield a higher numerical accuracy and a longer computation time. However, there is an upper limit where a significant increase in elements yields only minimal increases in computational accuracy while still greatly increasing computation time. The studies performed in this section show results where the number of cross-sectional elements in a center-gated disk was varied until little increase in accuracy was observed. Any further number of elements over 13 through the cross-section of the mold yielded differences in predicted velocity fields of < 0.12 %.

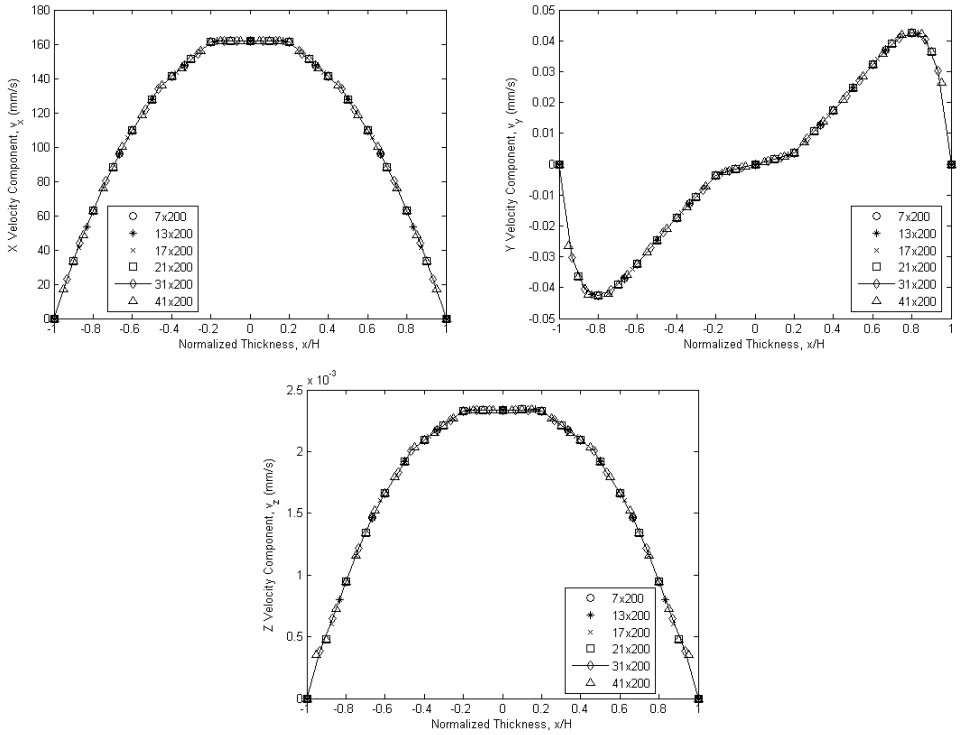

Figure B.5: Results for (Up-Left) x-velocity (Up-Right) y-velocity (Bottom) z-velocity components during mesh benchmarking. The number of cross-sectional elements is varied from 7 to 41.

One major drawback to the ANSYS Polyflow solver is the limits that the program puts on interpolation between elements. Using the volume of fluid method in Polyflow limits the interpolation between elements to a linear basis function only. This is because the solver is designed to handle the solution of complicated viscoelastic constitutive relations where a linear interpolation provides stability to the solution. For this reason, a higher number of elements (31 elements through the thickness of both the center-gated disk and end-gated plaque) were often chosen to try and minimize the effect of linear interpolation between elements.

B.3 Improvements to the FDM Solution Method

The majority of the computations for the work presented here have been performed through the use of MATLAB® (2011b, ver. 7.13.0.564, The Mathworks, Inc., Natick, MA) core language and a number of the associated built-in functions. A number of other language exist

which provide more efficient computations such as C and FORTRAN, however, MATLAB was chosen because of the ease of use and the results visualization ability that the language provides.

B.3.1 **Major Code Updates**

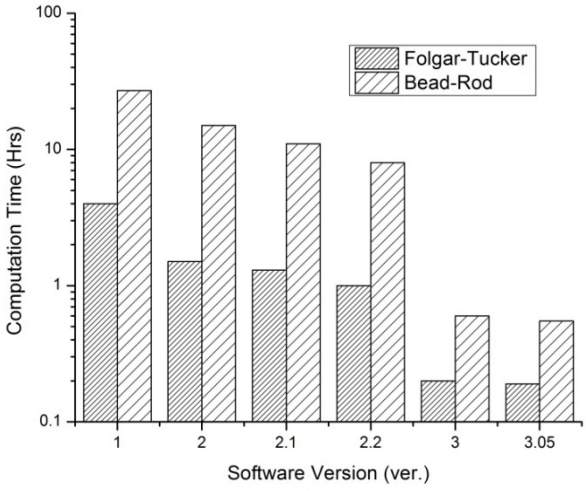

Figure B.6: Reduction of Computation Time with Successive Code Improvements. Version 2 introduced vectorization into the MATLAB routines. Version 3 was the conversion of some core routines into the C programming language.

The MATLAB code which is being used for the simulations presented in this work has undergone a number of revisions since the inception in early 2011. Figure B.6 above shows the improvements in computational time for two models in calculating the fiber orientation in a generic center-gated disk simulation domain. The first attempt at writing the software was made with convenience to the programmer in mind, so that all of the code was in matrix format (i.e the data was stored in m-by-n data arrays). The 2nd revision of code was the transformation of the matrix format to vectorized format in MATLAB (1-by-m data storage format) which resulted in the first major reduction of computation time. The 3rd major revision of the code involved rewriting some subroutines in the C programming language and calling these subroutines in the MATLAB environment through the "mex" file format. This conversion of subroutines allowed an order of magnitude reduction in computation time. The 4th major revision of the software will include a GUI interface so that users unfamiliar with higher level MATLAB/C programming will still be capable of using the developed software package with only marginal training.

B.4 References

[1] H. H. Winter, "Viscous dissipation term in energy equations," in *Calculation and Measurement Techniques for Momentum, Energy, and Mass Transfer*. vol. 7, R. J. Gordon, Ed., ed: AIChE Modular Instruction, 1987, pp. 27-34.

[2] J. R. A. Pearson, "Polymer Flows Dominated by High Heat-Generation and Low Heat-Transfer," *Polymer Engineering and Science,* vol. 18, pp. 222-229, 1978.

Appendix: Experimental Rheological Data

C Appendix: Rheological Data

C.1 Neat SABIC Polypropylene Data

C.1.1 Dynamic Neat Matrix Data at 180°C

180°C				
Frequency (rad/s)	G' (Pa)	G'' (Pa)	Torque (g*cm)	η*(Pa*s)
0.10	0.239	37.070	0.619	370.706
0.13	0.295	46.702	0.779	370.973
0.16	0.394	58.788	0.981	370.938
0.20	0.663	73.952	1.234	370.659
0.25	0.908	93.065	1.553	370.527
0.32	1.516	117.163	1.955	370.547
0.40	2.412	147.329	2.459	370.143
0.50	3.689	185.399	3.095	370.012
0.63	5.692	233.129	3.892	369.616
0.79	8.668	292.788	4.889	368.788
1.00	13.454	368.001	6.146	368.247
1.26	20.582	459.793	7.683	365.632
1.58	31.167	575.970	9.629	363.985
2.00	46.524	720.872	12.059	362.086
2.51	69.154	901.088	15.093	359.835
3.16	102.241	1122.870	18.835	356.605
3.98	151.019	1397.300	23.484	353.088
5.01	218.717	1731.070	29.160	348.201
6.31	313.426	2138.860	36.138	342.673
7.94	445.809	2633.770	44.662	336.358
10.00	623.813	3225.300	54.933	328.579
12.59	867.118	3934.620	67.377	320.109
15.85	1188.980	4773.880	82.258	310.486
19.95	1607.900	5757.740	99.908	299.685
25.11	2149.780	6906.730	120.761	288.049
31.61	2840.410	8228.290	145.017	275.343
39.80	3710.720	9735.650	172.873	261.781
50.10	4790.230	11448.900	204.440	247.696
63.08	6107.210	13362.000	238.944	232.913
79.41	7692.980	15472.000	275.137	217.597
100.00	9579.140	17794.500	310.779	202.090

Figure C.1: Frequency sweep data for SABIC Verton® polypropylene matrix at 5% strain in 25mm cone-and-plate fixture obtained from RMS-800 at 180°C.

C.1.2 Dynamic Neat Matrix Data at 190°C

190°C				
Frequency (rad/s)	G' (Pa)	G'' (Pa)	Torque (g*cm)	η*(Pa*s)
0.10	0.131	28.141	0.470	281.409
0.13	0.160	35.364	0.590	280.910
0.16	0.271	44.416	0.741	280.252
0.20	0.328	55.944	0.934	280.391
0.25	0.587	70.286	1.173	279.831
0.32	0.831	88.422	1.476	279.638
0.40	1.379	111.301	1.858	279.612
0.50	2.096	140.019	2.337	279.421
0.63	3.249	176.112	2.940	279.183
0.79	5.004	221.285	3.695	278.675
1.00	7.810	278.152	4.645	278.287
1.26	12.106	349.542	5.839	277.848
1.58	18.426	438.584	7.329	277.003
2.00	28.373	549.834	9.192	275.969
2.51	41.911	688.274	11.519	274.553
3.16	63.783	859.781	14.406	272.674
3.98	94.317	1074.160	18.022	270.900
5.01	139.932	1335.690	22.454	268.011
6.31	204.966	1655.740	27.903	264.472
7.94	293.908	2049.420	34.640	260.700
10.00	417.693	2522.170	42.790	255.652
12.59	566.489	2963.590	50.524	239.722
15.85	805.318	3528.670	60.613	228.422
19.95	1052.980	4007.440	69.406	207.716
25.11	1352.800	4686.810	81.693	194.253
31.61	1728.970	5454.890	95.721	181.005
39.80	2208.600	6477.370	114.187	171.949
50.10	2892.000	7846.720	138.763	166.905
63.08	3708.360	9160.930	162.331	156.682
79.41	4441.970	10032.800	177.260	138.174
100.00	4790.780	10168.300	177.220	112.403

Figure C.2: Frequency sweep data for SABIC Verton® polypropylene matrix at 5% strain in 25mm cone-and-plate fixture obtained from RMS-800 at 190°C.

C.1.3 Dynamic Neat Matrix Data at 200°C

200°C				
Frequency (rad/s)	G' (Pa)	G'' (Pa)	Torque (g*cm)	η*(Pa*s)
0.10	0.011	22.483	0.375	224.829
0.13	0.016	28.339	0.473	225.106
0.16	0.104	35.664	0.595	225.025
0.20	0.235	44.837	0.748	224.723
0.25	0.300	56.354	0.940	224.359
0.32	0.514	70.970	1.184	224.442
0.40	0.773	89.278	1.490	224.276
0.50	1.226	112.297	1.874	224.088
0.63	1.950	141.437	2.361	224.197
0.79	3.162	177.847	2.969	223.949
1.00	4.993	223.521	3.732	223.597
1.26	7.735	281.092	4.694	223.388
1.58	12.214	352.970	5.896	222.867
2.00	18.504	443.017	7.403	222.255
2.51	29.309	555.309	9.291	221.411
3.16	43.156	695.741	11.648	220.468
3.98	64.836	868.589	14.559	218.823
5.01	96.014	1082.110	18.167	216.796
6.31	142.520	1349.260	22.698	215.075
7.94	205.410	1672.720	28.208	212.208
10.00	298.382	2068.490	34.996	208.990
12.59	419.403	2533.900	43.039	204.059
15.85	590.171	3107.060	53.013	199.594
19.95	820.100	3790.230	65.026	194.405
25.11	1124.800	4603.910	79.456	188.725
31.61	1525.250	5558.220	96.527	182.313
39.80	2043.980	6666.000	116.486	175.185
50.10	2705.930	7947.390	139.523	167.558
63.08	3537.620	9407.730	165.376	159.343
79.41	4572.700	11053.900	193.449	150.644
100.00	5838.790	12896.800	222.422	141.569

Figure C.3: Frequency sweep data for SABIC Verton® polypropylene matrix at 5% strain in 25mm cone-and-plate fixture obtained from RMS-800 at 200°C.

C.1.4 Dynamic Neat Matrix Data at 220°C

220°C				
Frequency (rad/s)	G' (Pa)	G'' (Pa)	Torque (g*cm)	η*(Pa*s)
0.10	0.033	15.823	0.264	158.230
0.13	0.048	19.834	0.331	157.543
0.16	0.143	24.880	0.415	156.986
0.20	0.112	31.334	0.523	157.046
0.25	0.170	39.372	0.657	156.749
0.32	0.251	49.502	0.826	156.548
0.40	0.407	62.369	1.041	156.676
0.50	0.650	78.386	1.308	156.414
0.63	0.926	98.695	1.648	156.437
0.79	1.515	124.133	2.073	156.298
1.00	2.497	156.155	2.607	156.189
1.26	3.881	196.397	3.280	156.051
1.58	6.042	246.850	4.123	155.816
2.00	9.407	310.060	5.180	155.488
2.51	14.012	388.805	6.501	154.908
3.16	22.174	488.016	8.166	154.507
3.98	33.632	612.359	10.255	154.075
5.01	51.484	765.727	12.838	153.155
6.31	76.131	955.648	16.045	151.969
7.94	113.909	1192.010	20.054	150.779
10.00	165.951	1481.270	24.978	149.054
12.59	238.069	1815.970	30.716	145.515
15.85	340.707	2242.170	38.060	143.129
19.95	481.542	2755.370	46.971	140.223
25.11	674.577	3374.930	57.807	137.052
31.61	932.588	4108.620	70.727	133.267
39.80	1272.380	4976.180	86.056	129.053
50.10	1717.740	5989.920	103.925	124.367
63.08	2292.800	7163.420	124.343	119.241
79.41	3014.580	8508.280	146.960	113.673
100.00	3924.440	10033.800	170.986	107.740

Figure C.4: Frequency sweep data for SABIC Verton® polypropylene matrix at 5% strain in 25mm cone-and-plate fixture obtained from RMS-800 at 220°C.

C.1.5 Dynamic Viscosity Data

Frequency, ω (s^{-1})	Dynamic Viscosity, η (Pa*s)			
	180°C	190°C	200°C	220°C
0.100	371.56	293.03	227.99	153.90
0.126	371.52	292.58	226.27	153.72
0.158	371.12	292.80	225.63	153.46
0.200	371.07	292.75	225.21	153.31
0.251	370.74	292.25	224.76	153.14
0.316	370.71	292.35	224.76	153.03
0.398	369.88	292.18	224.50	154.41
0.501	369.33	292.02	224.34	154.24
0.631	368.95	291.82	224.34	154.11
0.794	368.12	291.38	223.09	153.94
1.000	367.37	292.00	222.99	152.84
1.259	365.80	291.37	222.32	152.74
1.585	363.99	290.39	221.88	152.45
1.995	362.07	289.12	221.27	152.19
2.512	359.64	287.67	220.38	151.73
3.162	356.61	285.57	219.36	151.34
3.980	352.84	283.12	217.98	150.72
5.011	348.05	280.02	216.10	149.93
6.308	342.61	276.25	213.94	148.85
7.942	336.23	272.00	211.35	147.64
9.998	328.53	266.76	208.03	145.97
12.586	320.07	262.07	203.93	142.86
15.845	310.40	255.11	199.49	140.73
19.948	299.58	247.17	194.33	137.91
25.112	287.94	238.53	188.64	134.80
31.614	275.23	229.05	182.20	131.20
39.800	261.71	218.82	175.11	127.06
50.105	247.60	207.98	167.50	122.53
63.077	232.81	196.52	159.27	117.54
79.408	217.50	184.64	150.59	112.09
100.000	202.01	172.29	141.51	106.28

Figure C.5: Frequency sweep data for SABIC Verton® polypropylene matrix at 5% strain in 25mm cone-and-plate fixture obtained from RMS-800.

C.1.6 Neat Matrix Capillary Data

180°C		220°C	
Shear Rate, γ (s^{-1})	Viscosity, η (Pa*s)	Shear Rate, γ (s^{-1})	Viscosity, η (Pa*s)
18.49	343.90	18.10	201.65
36.98	225.45	36.20	143.30
92.44	201.26	90.50	117.33
184.89	154.65	181.01	91.44
369.78	102.30	362.01	63.63
924.44	70.31	905.03	39.96
1848.88	48.83	1810.06	26.73
3697.76	29.21	3620.12	19.13
9244.40	13.87	9050.31	10.46
18488.80	8.21	18100.61	6.29
36977.60	4.76	36201.22	3.78

Figure C.6: Capillary data for SABIC Verton® polypropylene Matrix from Göttfert Capillary Rheometer.

C.2 SABIC Polypropylene w/ 30 % (wt.) Long Glass Fiber

Time (s)	Transient Viscosity, η^+ (Pa*s)		
	$\gamma = 0.4\ s^{-1}$	$\gamma = 1.0\ s^{-1}$	$\gamma = 4.0\ s^{-1}$
0	0.00	0.00	0.00
2	4370.93	542.47	320.59
4	5325.34	592.90	391.13
6	5839.11	653.81	397.75
8	6517.87	696.19	353.14
10	7138.55	713.05	293.43
12	7284.88	801.96	262.55
14	7416.44	819.57	236.34
16	7489.17	844.13	272.91
18	7524.45	954.63	265.78
20	7534.63	964.52	225.36
22	7527.77	919.72	276.60
24	7514.05	993.22	410.22
26	7433.03	1025.18	524.78
28	7346.26	1041.97	509.18
30	7251.72	1091.90	524.40
32	7131.99	1041.87	523.62
34	6989.76	1022.83	634.46
36	6844.88	1122.28	811.64
38	6706.05	1056.02	912.01
40	6569.05	894.59	1016.10
42	6404.43	1016.89	1075.12
44	6239.81	1114.16	1089.26
46	6107.48	1067.42	1100.68
48	6002.37	894.94	1084.18
50	5921.79	921.02	981.35
52	5846.96	1078.23	885.63
54	5760.91	1017.11	739.55
56	5673.13	933.08	682.65
58	5590.82	938.22	698.13
60	5522.90	861.13	698.79
62	5478.00	738.98	644.56
64	5424.09	812.47	606.40
66	5372.38	805.74	638.11
68	5328.48	792.00	682.16
70	5293.23	770.16	0.00
72	5281.47	714.39	0.00
74	5263.69	773.64	0.00
76	5236.25	784.35	0.00
78	5229.92	1073.69	0.00
80	5227.81	952.89	0.00
82	5214.09	1040.45	0.00
84	5217.61	1029.53	0.00
86	5229.92	897.04	0.00
88	5229.92	788.99	0.00
90	5215.99	756.78	0.00
92	5207.76	687.21	0.00
94	5200.97	676.83	0.00
96	5177.46	704.29	0.00
98	5170.82	683.94	0.00
100	5185.60	640.63	0.00
102	5185.60	559.49	0.00
104	5156.31	416.60	0.00
106	5141.27	607.01	0.00
108	5135.47	588.95	0.00
110	5121.75	500.17	0.00
112	5119.11	349.04	0.00
114	5114.63	461.52	0.00
116	5100.91	557.33	0.00
118	5096.95	624.83	0.00
120	5096.95	321.24	0.00

Figure C.7: Transient viscosity data for SABIC Verton® polypropylene/ 30 wt. % LGF obtained from sliding plate rheometer at 180° C

C.3 Neat RTP Polypropylene Data

C.3.1 Dynamic Neat Matrix Data at 180°C

180°C			
Frequency (rad/s)	G' (Pa)	G'' (Pa)	η*(Pa*s)
0.10	154.67	745.30	7612.81
0.13	203.99	905.40	7373.22
0.16	267.39	1097.36	7127.66
0.20	350.40	1325.31	6871.77
0.25	456.39	1595.05	6606.19
0.32	590.84	1914.36	6336.82
0.40	760.42	2288.29	6058.34
0.50	975.28	2724.05	5774.41
0.63	1241.30	3227.30	5481.60
0.79	1574.39	3810.95	5192.40
1.00	1979.97	4477.93	4897.48
1.26	2475.02	5233.23	4599.64
1.58	3071.99	6089.24	4304.52
1.99	3788.63	7051.29	4013.04
2.51	4641.12	8109.05	3720.78
3.16	5655.24	9295.42	3441.85
3.98	6852.23	10602.49	3172.07
5.01	8221.02	12001.65	2903.58
6.31	9824.26	13545.30	2652.97
7.94	11671.44	15192.99	2412.82
10.00	13762.02	16955.54	2184.62
12.58	16129.87	18841.93	1970.96
15.84	18815.42	20797.32	1770.26
19.94	21804.95	22872.03	1584.43
25.11	25118.42	24992.24	1411.26
31.61	28748.82	27214.89	1252.43
39.79	32751.90	29471.49	1107.25
50.09	37107.38	31765.55	975.09
63.06	41797.39	34074.57	855.11
79.39	46828.77	36358.73	746.75
100.00	52162.53	38547.28	648.60

Figure C.8: Frequency sweep data for RTP 100® polypropylene matrix at 5% strain in 25mm cone-and-plate fixture obtained from RMS-800 at 180°C.

C.3.2 Dynamic Neat Matrix Data at 200°C

200°C			
Frequency (rad/s)	G' (Pa)	G'' (Pa)	η*(Pa*s)
0.10	75.55	451.21	4575.54
0.13	101.10	551.90	4457.52
0.16	135.24	673.87	4337.36
0.20	179.51	820.23	4208.97
0.25	237.88	995.82	4076.81
0.32	313.04	1205.56	3939.55
0.40	408.92	1453.71	3794.13
0.50	530.73	1746.72	3643.36
0.63	686.79	2089.16	3486.30
0.79	880.90	2490.43	3326.51
1.00	1125.13	2953.74	3161.64
1.26	1425.76	3487.63	2993.71
1.58	1796.17	4101.79	2826.12
1.99	2249.31	4799.89	2657.49
2.51	2789.74	5585.00	2486.15
3.16	3457.12	6474.23	2321.68
3.98	4240.64	7471.60	2158.72
5.01	5174.65	8547.72	1994.36
6.31	6281.85	9745.42	1838.30
7.94	7542.63	11052.77	1685.22
10.00	9018.84	12471.61	1539.69
12.58	10725.71	13985.75	1400.57
15.84	12652.42	15598.18	1267.76
19.94	14846.68	17313.01	1143.54
25.11	17328.53	19116.24	1027.62
31.61	20080.53	20993.31	919.08
39.79	23135.15	22939.81	818.76
50.09	26503.45	24924.20	726.26
63.06	30169.29	26934.89	641.30
79.39	34164.58	28945.09	564.00
100.00	38446.43	30913.48	493.33

Figure C.9: Frequency sweep data for RTP 100® polypropylene matrix at 5% strain in 25mm cone-and-plate fixture obtained from RMS-800 at 200°C.

C.3.3 Dynamic Neat Matrix Data at 220°C

220°C			
Frequency (rad/s)	G' (Pa)	G'' (Pa)	η*(Pa*s)
0.10	91.72	261.40	2770.63
0.13	105.90	316.98	2655.05
0.16	122.75	385.56	2553.44
0.20	143.88	469.05	2459.39
0.25	172.91	569.46	2369.76
0.32	208.49	691.79	2285.30
0.40	254.91	838.72	2202.41
0.50	313.82	1014.15	2118.67
0.63	390.23	1223.64	2036.08
0.79	487.86	1474.25	1955.48
1.00	610.66	1766.21	1869.31
1.26	766.55	2112.55	1785.61
1.58	964.32	2514.95	1699.96
1.99	1210.33	2981.13	1613.04
2.51	1520.93	3518.85	1526.61
3.16	1888.83	4137.28	1438.69
3.98	2345.02	4840.13	1351.42
5.01	2894.66	5626.09	1262.86
6.31	3564.84	6519.79	1178.12
7.94	4361.73	7502.53	1092.94
10.00	5311.42	8589.85	1010.33
12.58	6415.75	9779.23	929.41
15.84	7699.91	11082.70	851.82
19.94	9183.40	12494.40	777.48
25.11	10902.62	14002.76	706.82
31.61	12845.05	15614.37	639.67
39.79	15047.85	17332.39	576.83
50.09	17528.94	19110.99	517.67
63.06	20263.37	20955.37	462.23
79.39	23298.78	22846.00	411.01
100.00	26620.33	24753.03	363.50

Figure C.10: Frequency sweep data for RTP 100® polypropylene matrix at 5% strain in 25mm cone-and-plate fixture obtained from RMS-800 at 220°C.

Appendix: Experimental Fiber Orientation Data

D Appendix: Fiber Orientation Data

D.1 Center-Gated Disk LGF Orientation Data – 0.7mm Bin Width

0%						
z/2H	**A11**	**A12**	**A22**	**A33**	**A13**	**A23**
0.92	0.56	0.14	0.10	0.34	0.29	0.08
0.77	0.73	0.18	0.09	0.18	0.26	0.06
0.62	0.60	0.15	0.07	0.33	0.32	0.06
0.46	0.40	0.06	0.07	0.53	0.36	0.05
0.31	0.47	0.08	0.05	0.48	0.33	0.06
0.15	0.35	0.05	0.05	0.61	0.35	0.04
0.00	0.27	0.00	0.07	0.67	0.35	0.04
-0.15	0.41	-0.04	0.04	0.55	0.38	-0.04
-0.31	0.44	-0.05	0.03	0.53	0.40	-0.06
-0.46	0.38	-0.02	0.02	0.59	0.38	-0.04
-0.62	0.57	-0.04	0.02	0.41	0.31	-0.02
-0.77	0.41	-0.05	0.04	0.55	0.37	-0.07
-0.92	0.48	0.00	0.03	0.50	0.35	-0.01

Figure D.1: Experimentally determined orientation values at 0% fill of the center-gated disk for SABIC Verton PP/30 wt. % LGF using bin width of 0.7 mm.

10%						
z/2H	**A11**	**A12**	**A22**	**A33**	**A13**	**A23**
0.92	0.55	0.04	0.02	0.42	0.38	0.04
0.77	0.46	0.02	0.04	0.50	0.36	-0.01
0.62	0.58	0.04	0.04	0.38	0.32	0.03
0.46	0.36	0.01	0.05	0.59	0.33	0.03
0.31	0.24	0.01	0.03	0.72	0.29	0.02
0.15	0.24	-0.01	0.04	0.72	0.35	-0.02
0.00	0.43	-0.01	0.03	0.54	0.37	0.01
-0.15	0.49	-0.04	0.04	0.47	0.34	-0.01
-0.31	0.35	-0.02	0.03	0.62	0.36	-0.01
-0.46	0.39	-0.03	0.04	0.57	0.32	-0.01
-0.62	0.33	-0.03	0.04	0.63	0.33	-0.05
-0.77	0.49	0.01	0.03	0.48	0.36	0.01
-0.92	0.37	0.00	0.04	0.59	0.36	-0.03

Figure D.2: Experimentally determined orientation values at 10% fill of the center-gated disk for SABIC Verton PP/30 wt. % LGF using bin width of 0.7 mm.

40%						
z/2H	**A11**	**A12**	**A22**	**A33**	**A13**	**A23**
0.92	0.49	0.02	0.03	0.48	0.33	0.03
0.77	0.52	0.00	0.05	0.43	0.32	0.02
0.62	0.53	0.01	0.05	0.42	0.33	0.02
0.46	0.43	0.03	0.05	0.52	0.34	0.01
0.31	0.26	0.01	0.05	0.69	0.27	0.01
0.15	0.20	0.01	0.06	0.74	0.30	0.01
0.00	0.31	0.00	0.05	0.64	0.35	-0.02
-0.15	0.36	-0.04	0.04	0.60	0.34	-0.05
-0.31	0.34	-0.03	0.05	0.61	0.30	-0.03
-0.46	0.37	-0.01	0.04	0.59	0.34	-0.05
-0.62	0.42	-0.01	0.04	0.54	0.37	-0.03
-0.77	0.39	-0.02	0.04	0.57	0.37	-0.04
-0.92	0.27	-0.01	0.05	0.69	0.33	-0.03

Figure D.3: Experimentally determined orientation values at 40% fill of the center-gated disk for SABIC Verton PP/30 wt. % LGF using bin width of 0.7 mm.

90%						
z/2H	**A11**	**A12**	**A22**	**A33**	**A13**	**A23**
0.92	0.46	0.04	0.07	0.47	0.35	0.03
0.77	0.44	0.00	0.08	0.48	0.34	0.02
0.62	0.37	0.03	0.09	0.54	0.35	0.02
0.46	0.32	0.02	0.08	0.60	0.35	0.02
0.31	0.30	0.01	0.07	0.63	0.28	0.00
0.15	0.17	0.01	0.06	0.77	0.31	0.01
0.00	0.16	0.01	0.07	0.78	0.27	0.03
-0.15	0.28	0.00	0.05	0.67	0.36	0.00
-0.31	0.35	-0.01	0.04	0.61	0.36	-0.01
-0.46	0.35	0.01	0.04	0.61	0.37	0.03
-0.62	0.34	0.01	0.04	0.62	0.34	0.00
-0.77	0.32	-0.01	0.03	0.65	0.37	-0.02
-0.92	0.48	0.00	0.04	0.48	0.35	0.01

Figure D.4: Experimentally determined orientation values at 40% fill of the center-gated disk for SABIC Verton PP/30 wt. % LGF using bin width of 0.7 mm.

D.2 End-Gated Plaque Mold LGF Orientation Data – 0.7 mm Bin Width

0% Length, 0% Width						
z/2H	**A11**	**A12**	**A22**	**A33**	**A13**	**A23**
0.92	0.60	0.09	0.05	0.35	0.07	0.06
0.77	0.38	0.05	0.06	0.56	0.04	0.11
0.62	0.30	0.15	0.17	0.53	0.19	0.12
0.46	0.37	0.12	0.07	0.56	0.13	0.10
0.31	0.41	0.15	0.09	0.50	0.18	0.10
0.15	0.36	0.14	0.10	0.54	0.12	0.12
0.00	0.23	0.11	0.08	0.69	0.10	0.13
-0.15	0.24	0.07	0.07	0.69	0.11	0.10
-0.31	0.18	0.05	0.08	0.75	0.08	0.07
-0.46	0.11	0.02	0.04	0.84	-0.01	0.04
-0.62	0.21	0.00	0.02	0.77	0.02	0.01
-0.77	0.18	-0.02	0.03	0.80	-0.04	0.01
-0.92	0.64	-0.05	0.02	0.34	-0.19	-0.01

Figure D.5: Experimentally determined orientation values at 0% fill and 0% of width in the end-gated plaque for SABIC Verton PP/30 wt. % LGF using bin width of 0.7 mm.

10% Length, 0% Width						
z/2H	**A11**	**A12**	**A22**	**A33**	**A13**	**A23**
0.92	0.40	0.04	0.06	0.54	0.01	0.04
0.77	0.53	-0.01	0.06	0.41	0.10	0.02
0.62	0.47	0.02	0.05	0.48	0.10	0.03
0.46	0.17	0.00	0.09	0.74	0.04	-0.02
0.31	0.29	0.05	0.09	0.62	0.09	0.08
0.15	0.32	0.04	0.11	0.58	0.06	0.07
0.00	0.44	-0.01	0.08	0.48	0.04	0.02
-0.15	0.21	0.02	0.18	0.61	0.01	0.07
-0.31	0.19	-0.02	0.19	0.62	0.03	-0.03
-0.46	0.18	0.04	0.18	0.64	-0.03	0.10
-0.62	0.37	-0.01	0.17	0.46	-0.03	0.05
-0.77	0.51	0.03	0.11	0.39	0.01	0.06
-0.92	0.53	0.00	0.02	0.44	0.08	0.01

Figure D.6: Experimentally determined orientation values at 10% fill and 0% of width in the end-gated plaque for SABIC Verton PP/30 wt. % LGF using bin width of 0.7 mm.

40% Length, 0% Width						
z/2H	**A11**	**A12**	**A22**	**A33**	**A13**	**A23**
0.92	0.53	0.00	0.02	0.45	0.05	0.03
0.77	0.43	0.01	0.02	0.55	0.16	0.02
0.62	0.36	0.01	0.04	0.60	0.10	0.06
0.46	0.45	0.03	0.05	0.50	0.10	0.03
0.31	0.35	0.02	0.02	0.62	0.14	0.03
0.15	0.28	0.06	0.04	0.68	0.17	0.08
0.00	0.25	0.04	0.04	0.72	0.12	0.06
-0.15	0.10	-0.01	0.04	0.86	0.08	0.01
-0.31	0.19	0.01	0.04	0.77	0.12	0.04
-0.46	0.40	-0.04	0.04	0.56	-0.01	0.07
-0.62	0.60	0.01	0.04	0.37	0.06	0.03
-0.77	0.52	-0.01	0.06	0.41	0.08	0.02
-0.92	0.53	0.01	0.03	0.44	0.10	0.02

Figure D.7: Experimentally determined orientation values at 40% fill and 0% of width in the end-gated plaque for SABIC Verton PP/30 wt. % LGF using bin width of 0.7 mm.

90% Length, 0% Width						
z/2H	**A11**	**A12**	**A22**	**A33**	**A13**	**A23**
0.92	0.51	0.07	0.04	0.45	0.05	0.08
0.77	0.69	0.04	0.01	0.30	-0.03	0.02
0.62	0.26	0.03	0.02	0.72	0.07	0.04
0.46	0.20	0.02	0.03	0.76	0.00	0.05
0.31	0.18	0.01	0.03	0.79	0.02	0.03
0.15	0.07	0.01	0.03	0.90	0.04	0.05
0.00	0.18	0.03	0.08	0.74	0.08	0.09
-0.15	0.19	0.10	0.11	0.70	0.04	0.05
-0.31	0.16	0.06	0.08	0.76	0.05	0.09
-0.46	0.18	0.05	0.10	0.72	0.00	0.07
-0.62	0.31	0.01	0.06	0.63	0.00	0.02
-0.77	0.20	-0.02	0.04	0.76	0.00	0.00
-0.92	0.71	-0.03	0.03	0.27	0.02	0.01

Figure D.8: Experimentally determined fiber orientation values at 90% fill and 0% of width in the end-gated plaque for SABIC Verton PP/30 wt. % LGF using bin width of 0.7 mm.

0% Length, 50% Width						
z/2H	**A11**	**A12**	**A22**	**A33**	**A13**	**A23**
0.92	0.54	0.12	0.07	0.39	0.25	0.08
0.77	0.59	0.12	0.05	0.36	-0.02	0.07
0.62	0.54	0.11	0.05	0.41	0.04	0.06
0.46	0.55	0.07	0.04	0.41	0.03	0.06
0.31	0.45	0.12	0.06	0.50	0.15	0.10
0.15	0.57	0.12	0.05	0.38	0.16	0.08
0.00	0.70	0.14	0.05	0.25	0.24	0.07
-0.15	0.64	0.12	0.04	0.32	0.15	0.06
-0.31	0.69	0.12	0.04	0.27	0.19	0.06
-0.46	0.45	0.04	0.03	0.52	0.02	0.05
-0.62	0.38	0.05	0.03	0.59	0.06	0.05
-0.77	0.53	0.00	0.02	0.45	-0.09	0.01
-0.92	0.50	-0.01	0.02	0.48	-0.11	0.01

Figure D.9: Experimentally determined orientation values at 0% fill and 50% of width in the end-gated plaque for SABIC Verton PP/30 wt. % LGF using bin width of 0.7 mm.

10% Length, 50% Width						
z/2H	**A11**	**A12**	**A22**	**A33**	**A13**	**A23**
0.92	0.53	0.01	0.01	0.46	0.06	0.02
0.77	0.64	-0.01	0.02	0.35	0.01	0.03
0.62	0.66	0.02	0.02	0.32	0.05	0.03
0.46	0.75	0.02	0.01	0.23	0.03	0.02
0.31	0.55	0.03	0.01	0.44	0.06	0.02
0.15	0.59	0.04	0.02	0.39	0.09	0.03
0.00	0.65	0.03	0.02	0.33	0.07	0.03
-0.15	0.66	0.00	0.01	0.32	0.01	0.03
-0.31	0.57	0.01	0.03	0.41	0.02	0.03
-0.46	0.53	-0.02	0.02	0.44	-0.01	0.03
-0.62	0.51	-0.01	0.01	0.48	-0.03	0.02
-0.77	0.57	0.00	0.02	0.41	-0.03	0.03
-0.92	0.57	0.00	0.01	0.42	-0.01	0.03

Figure D.10: Experimentally determined orientation values at 10% fill and 50% of width in the end-gated plaque for SABIC Verton PP/30 wt. % LGF using bin width of 0.7 mm.

40% Length, 50% Width						
z/2H	**A11**	**A12**	**A22**	**A33**	**A13**	**A23**
0.92	0.41	0.01	0.02	0.56	0.02	0.03
0.77	0.45	0.00	0.02	0.54	-0.01	0.03
0.62	0.71	0.01	0.02	0.27	0.03	0.03
0.46	0.75	0.01	0.02	0.23	0.04	0.03
0.31	0.71	0.02	0.01	0.27	0.02	0.02
0.15	0.57	0.01	0.02	0.41	0.04	0.04
0.00	0.62	0.04	0.03	0.36	0.03	0.04
-0.15	0.63	0.02	0.03	0.34	0.03	0.04
-0.31	0.50	-0.01	0.02	0.48	-0.01	0.04
-0.46	0.58	-0.01	0.01	0.40	-0.02	0.03
-0.62	0.54	-0.01	0.02	0.45	-0.03	0.04
-0.77	0.69	-0.01	0.02	0.29	-0.02	0.03
-0.92	0.71	-0.01	0.02	0.28	-0.03	0.03

Figure D.11: Experimentally determined orientation values at 40% fill and 50% of width in the end-gated plaque for SABIC Verton PP/30 wt. % LGF using bin width of 0.7 mm.

90% Length, 50% Width						
z/2H	**A11**	**A12**	**A22**	**A33**	**A13**	**A23**
0.92	0.41	0.02	0.03	0.56	0.08	0.04
0.77	0.46	0.01	0.01	0.52	0.01	0.03
0.62	0.49	0.01	0.02	0.49	0.01	0.03
0.46	0.53	0.01	0.03	0.44	0.02	0.03
0.31	0.64	-0.02	0.02	0.35	-0.03	0.03
0.15	0.64	0.01	0.04	0.32	-0.01	0.03
0.00	0.63	0.04	0.03	0.35	0.02	0.03
-0.15	0.58	0.01	0.02	0.40	0.02	0.03
-0.31	0.62	0.00	0.04	0.34	-0.03	0.04
-0.46	0.62	0.00	0.02	0.36	-0.02	0.04
-0.62	0.65	-0.01	0.02	0.33	-0.01	0.04
-0.77	0.64	0.02	0.02	0.34	0.06	0.03
-0.92	0.65	0.03	0.02	0.32	0.10	0.04

Figure D.12: Experimentally determined orientation values at 90% fill and 50% of width in the end-gated plaque for SABIC Verton PP/30 wt. % LGF using bin width of 0.7 mm.

0% Length, 90% Width						
z/2H	**A11**	**A12**	**A22**	**A33**	**A13**	**A23**
0.92	0.71	0.16	0.06	0.23	0.12	0.05
0.77	0.78	0.18	0.05	0.16	0.27	0.06
0.62	0.77	0.12	0.05	0.18	0.23	0.05
0.46	0.60	0.12	0.05	0.36	0.12	0.06
0.31	0.74	0.14	0.07	0.19	0.18	0.05
0.15	0.67	0.17	0.07	0.26	0.19	0.05
0.00	0.67	0.15	0.06	0.27	0.12	0.08
-0.15	0.68	0.16	0.06	0.25	0.15	0.09
-0.31	0.78	0.09	0.03	0.19	0.15	0.04
-0.46	0.58	0.04	0.05	0.37	-0.01	0.04
-0.62	0.82	0.00	0.02	0.16	-0.07	0.01
-0.77	0.60	0.02	0.06	0.34	-0.12	0.06
-0.92	0.42	0.02	0.03	0.55	0.19	0.03

Figure D.13: Experimentally determined orientation values at 0% fill and 90% of width in the end-gated plaque for SABIC Verton PP/30 wt. % LGF using bin width of 0.7 mm.

10% Length, 90% Width						
z/2H	**A11**	**A12**	**A22**	**A33**	**A13**	**A23**
0.92	0.42	0.05	0.03	0.55	0.12	0.04
0.77	0.65	0.04	0.01	0.34	0.11	0.02
0.62	0.75	0.06	0.02	0.23	0.08	0.02
0.46	0.64	0.05	0.04	0.32	0.08	0.03
0.31	0.76	0.04	0.03	0.21	0.09	0.04
0.15	0.85	0.04	0.02	0.13	0.05	0.02
0.00	0.82	0.02	0.02	0.15	0.04	0.03
-0.15	0.81	0.04	0.03	0.16	0.11	0.04
-0.31	0.76	0.06	0.02	0.21	0.11	0.04
-0.46	0.70	0.00	0.03	0.27	-0.01	0.05
-0.62	0.69	0.01	0.03	0.28	-0.04	0.04
-0.77	0.63	-0.01	0.03	0.35	-0.07	0.04
-0.92	0.75	-0.02	0.01	0.24	-0.12	0.02

Figure D.14: Experimentally determined orientation values at 10% fill and 90% of width in the end-gated plaque for SABIC Verton PP/30 wt. % LGF using bin width of 0.7 mm.

40% Length, 90% Width						
z/2H	**A11**	**A12**	**A22**	**A33**	**A13**	**A23**
0.92	0.43	0.02	0.01	0.55	0.06	0.03
0.77	0.61	0.03	0.02	0.37	0.01	0.03
0.62	0.59	0.00	0.02	0.38	0.02	0.03
0.46	0.62	0.01	0.03	0.35	0.02	0.04
0.31	0.64	0.02	0.04	0.32	0.00	0.04
0.15	0.64	-0.01	0.03	0.33	-0.05	0.04
0.00	0.78	-0.02	0.02	0.20	-0.06	0.03
-0.15	0.74	-0.02	0.04	0.22	-0.05	0.04
-0.31	0.83	-0.01	0.03	0.14	-0.02	0.03
-0.46	0.76	0.00	0.02	0.22	-0.01	0.03
-0.62	0.77	-0.01	0.03	0.20	0.01	0.04
-0.77	0.68	-0.01	0.02	0.30	-0.06	0.03
-0.92	0.35	-0.01	0.01	0.64	-0.04	0.02

Figure D.15: Experimentally determined orientation values at 40% fill and 90% of width in the end-gated plaque for SABIC Verton PP/30 wt. % LGF using bin width of 0.7 mm.

90% Length, 90% Width						
z/2H	**A11**	**A12**	**A22**	**A33**	**A13**	**A23**
0.92	0.61	0.01	0.02	0.37	0.02	0.03
0.77	0.64	0.01	0.04	0.33	0.00	0.04
0.62	0.70	0.02	0.03	0.27	0.02	0.04
0.46	0.66	-0.01	0.02	0.31	-0.03	0.03
0.31	0.64	0.01	0.03	0.33	0.03	0.04
0.15	0.66	0.04	0.03	0.31	0.08	0.04
0.00	0.73	0.03	0.04	0.23	0.03	0.04
-0.15	0.76	0.05	0.03	0.21	0.06	0.04
-0.31	0.72	0.06	0.05	0.24	0.07	0.05
-0.46	0.63	0.05	0.06	0.31	0.03	0.06
-0.62	0.58	0.03	0.05	0.37	0.01	0.05
-0.77	0.61	0.00	0.02	0.37	0.00	0.03
-0.92	0.69	0.00	0.01	0.30	0.00	0.02

Figure D.16: Experimentally determined orientation values at 90% fill and 90% of width in the end-gated plaque for SABIC Verton PP/30 wt. % LGF using bin width of 0.7 mm.

E MATLAB Fiber Orientation Software Package

E.1 Fiber Orientation Interface Scripts

E.1.1 cgd_ft.m

```
%% Transient Cross-Section Orientation Calculations
% K.J. Meyer
% 12/10/2011

% clear all
% close all
% clc

%% Defining the POLYFLOW Files Directory
% This is the directory where the POLYFLOW export files are. The files
% should be '.txt' or '.csv' format with a ',' delimiter.
dir = 'D:\Computational Files\Transient CGD Simulation Files\PlanarDataRefine_2mm\';
type = '.csv';
timesteps = dlmread(sprintf('%s%s%s',dir,'timesteps','.txt'),',');



%% Defining the Constants for the FT Simulation
% Folgar-Tucker Constants
C_I = 0.003;
OF = 0.03;



%% Defining Additional Simulation Parameters
SendNotification = 0;
EmailAddress = 'kjmeyer@vt.edu';
save = 0; %Do you want to save the generated files in a txt format?
start = 6;
finish = 590;
fill = 299;
step = 5;

%% Pre-Allocation
% We pre-define a number of variables that would ususally grow inside a
% loop and slow down computation.
vx = zeros(31,fill);
vy = zeros(31,fill);
vz = zeros(31,fill);

fluid_frac = zeros(31,fill);

Aft_t2 = zeros(31,9,fill);
Aft_t1 = zeros(31,9,fill);

count = start;



%% Begin the Simulation

while (count <= finish) % Number of timesteps
   disp(count)
   data = dlmread(sprintf('%s%i%s',dir,count,type),',');
```

```
polydata = sortrows(data, [3 1 2]);

% Specifying the integration time, based on time input given
% above

for i = 2:fill
   % Number of fill percentages. Subtracting first and last step.
   % Numerical approximations are used and need i+1 and i-1 data.
   % These are not availible for 1 and 100.
   %=========================================================================%
   % To calculate the change in distance in the flow direction in the BR
   % model we need to load the cross-section data from the previous
   % two positions. Special conditions are needed for the first two
   % iterative steps in the solution because i /= 0;
   %=========================================================================%



   %=========================================================================%
   % Reading in the data for the cross-section of interest. For some
   % of the calculation we need one and two steps back from the region
   % of interest to estimate the derivative.
   %=========================================================================%
   pointm1 = (i-1)*31;
   point = i*31;
   pointp1 = (i+1)*31;

   v = [polydata(point-30:point,6) polydata(point-30:point,10) polydata(point-30:point,14)];
   v_m1 = [polydata(pointm1-30:pointm1,6) polydata(pointm1-30:pointm1,10) polydata(pointm1-30:pointm1,14)];
   v_p1 = [polydata(pointp1-30:pointp1,6) polydata(pointp1-30:pointp1,10) polydata(pointp1-30:pointp1,14)];

   % For visualization purposes

   vx(:,i) = v(:,1);
   vy(:,i) = v(:,2);
   vz(:,i) = v(:,3);

   dvi_dxi = zeros(31,9,fill);

   dvi_dxi(:,1,i) = polydata(point-30:point,7);
   dvi_dxi(:,2,i) = polydata(point-30:point,8);
   dvi_dxi(:,3,i) = polydata(point-30:point,9);
   dvi_dxi(:,4,i) = polydata(point-30:point,11);
   dvi_dxi(:,5,i) = polydata(point-30:point,12);
   dvi_dxi(:,6,i) = polydata(point-30:point,13);
   dvi_dxi(:,7,i) = polydata(point-30:point,15);
   dvi_dxi(:,8,i) = polydata(point-30:point,16);
   dvi_dxi(:,9,i) = polydata(point-30:point,17);

   fluid_frac(:,i) = polydata(point-30:point,4);



   %% ========================================================================%
   %                  Cross - Section Integration                  %
   %=========================================================================%
   % Here we integrate through each cross-section of data outputted by
   % POLYFLOW. j = 1:31 because we have chosen 31 cross-section
   % sampling points but the last sample is out of the domain
   % resulting in NAN's so we have neglected that.
```

```
%=========================================================================%

% Calculating the orientation values at different cross-sections
% (j = 1 to # of cross-sections).

for j = 1:31 % Number of cross sections

   % Inital orientation guesses, starting at position 2 (which is
   % really equal to 1% fill). Further iterations in the i
   % direction use previous solutions of orientation.

   if (fluid_frac(j,i) < 0.1)
      continue
   end

   if (i == 2)
      Aft_t1(j,:,1) = FT_initial(j);
      Aft_t1(j,:,i) = FT_initial(j);
   end

   % To estimate some of the derivatives in the models, we need to
   % calculate the change in coordinates. This calculates that
   % change as we iterate through (j,i) steps (thickness,fill).

   dX = [0.2983 0.0667 0];

   % Reading the velocity gradient data at each node of interest
   % and manipulating the data to calculate the vorticity and rate
   % of deformation.
   dvi = [dvi_dxi(j,1,i) dvi_dxi(j,2,i) dvi_dxi(j,3,i) dvi_dxi(j,4,i) dvi_dxi(j,5,i) dvi_dxi(j,6,i) dvi_dxi(j,7,i) dvi_dxi(j,8,i)
dvi_dxi(j,9,i)];
   delV = reshape(dvi,3,3);
   w = reshape((1/2)*(delV - delV'),1,9);
   d = reshape((1/2)*(delV + delV'),1,9);

   % If there is no previous orientation history, convect the
   % solution to the next spacial position.

   if ((j == 1) || (j == 31))
      A_0 = Aft_t1(j,:,i);

      if (A_0(1) == 0)
         A_0 = Aft_t2(j,:,i-1);
      end

      %          ftibof = @(t,A) ARD_RSC_IBOF_mex(w,d,OF,b1,b2,b3,b4,b5,A);
      ftibof = @(t,A) FT_IBOF_mex(w,d,C_I,OF,A);
      [t1,A1] = ode45(ftibof,linspace(timesteps(count-step),timesteps(count),5),A_0);
      Aft_t2(j,:,i) = A1(length(A1(:,1)),:);
      Aft_t2(j,:,i) = Aft_t2(j,:,i)./(Aft_t2(j,1,i) + Aft_t2(j,5,i) + Aft_t2(j,9,i));
      continue
   end

   if ((1 < j) && (j < 31))
      A_0 = Aft_t1(j,:,i);

      if (A_0(1) == 0)
```

```
            A_0 = Aft_t2(j,:,i-1);
        end

        Am11 = Aft_t2(j,:,i-1);
        Am12 = Aft_t2(j-1,:,i);
        Am13 = zeros(1,9);

        %           ftibof = @(t,A) ARD_RSC_IBOFc_mex(w,d,OF,b1,b2,b3,b4,b5,A,Am11,Am12,Am13,dX,[vx(j,i,k)
vy(j,i,k) vz(j,i,k)]);
        ftibof = @(t,A) FT_IBOFc_mex(w,d,C_I,OF,A,Am11,Am12,Am13,dX,[vx(j,i) vy(j,i) vz(j,i)]);
        [t1,A1] = ode45(ftibof,linspace(timesteps(count-step),timesteps(count),5),A_0);
        Aft_t2(j,:,i) = A1(length(A1(:,1)),:);
        Aft_t2(j,:,i) = Aft_t2(j,:,i)./(Aft_t2(j,1,i) + Aft_t2(j,5,i) + Aft_t2(j,9,i));
      end


    end
  end
  count = count + step;
  Aft_t1 = Aft_t2;
end
```

E.1.2 cgd_br.m

```
%% Transient Cross-Section Orientation Calculations
% K.J. Meyer
% 12/10/2011

clear all
close all
clc

%% Defining the POLYFLOW Files Directory
dir = 'D:\Computational Files\Transient CGD Simulation Files\PlanarDataRefine_2mm\';
type = '.csv';
timesteps = dlmread(sprintf('%s%s%s',dir,'timesteps','.txt'),',');

%% Defining the Constants for the Simulation
C_I = 0.053;
OF = 0.13;
lb = 1.8;
kbend = 218;

SendNotification = 0;
EmailAddress = 'kjmeyer@vt.edu';
save = 0; %Do you want to save the generated files in a txt format?
start = 6;
finish = 590;
fill = 299;
step = 5;



% Pre-allocation of internal variables for speed
A11br_cross = zeros(31,fill,finish);
A22br_cross = zeros(31,fill,finish);
A33br_cross = zeros(31,fill,finish);
A12br_cross = zeros(31,fill,finish);
A13br_cross = zeros(31,fill,finish);
vx = zeros(31,fill,finish);
```

```
vy = zeros(31,fill,finish);
vz = zeros(31,fill,finish);
fluid_frac = zeros(31,fill,finish);
temp = zeros(31,fill,finish);
xcoords = zeros(31,fill,finish);
ycoords = zeros(31,fill,finish);
zcoords = zeros(31,fill,finish);
dvi_dxi = zeros(31,9,fill,finish);

tbr = zeros(fill,finish);
Abr = zeros(31,9,fill,finish);
Bbr = zeros(31,9,fill,finish);
Cbr = zeros(31,3,fill,finish);
Rbr = zeros(31,9,fill,finish);

% Movie generation
% dos('del D:\Computational Files\Transient CGD Simulation Files\PlanarData2s\orient_quadconv.avi');
% mov = avifile([dir,'orient_quadconv.avi'],'Compression','none','fps',10);

%% Begin the Simulation
for k = start:step:finish % Number of timesteps
   data = dlmread(sprintf('%s%i%s',dir,k,type),',');
   polydata = sortrows(data, [1 2]);

   % Specifying the integration time, based on time input given
   % above

   % *********WARNING*********%
   % The time specification must be changed depending on the fill time
   % that is being studied. Remember to do this otherwise the predictions
   % will not converge.

   for i = 2:fill  % Number of fill percentages. Subtracting first and last step.
      % Numerical approximations are used and need i+1 and i-1 data.
      % These are not availible for 1 and 100.
      %=========================================================================%
      % To calculate the change in distance in the flow direction in the BR
      % model we need to load the cross-section data from the previous
      % two positions. Special conditions are needed for the first two
      % iterative steps in the solution because i /= 0;
      %=========================================================================%



      %=========================================================================%
      % Reading in the data for the cross-section of interest. For some
      % of the calculation we need one and two steps back from the region
      % of interest to estimate the derivative.
      %=========================================================================%
      pointm1 = (i-1)*31;
      point = i*31;
      pointp1 = (i+1)*31;

      v = [polydata(point-30:point,6) polydata(point-30:point,10) polydata(point-30:point,14)];
      v_m1 = [polydata(pointm1-30:pointm1,6) polydata(pointm1-30:pointm1,10) polydata(pointm1-30:pointm1,14)];
      v_p1 = [polydata(pointp1-30:pointp1,6) polydata(pointp1-30:pointp1,10) polydata(pointp1-30:pointp1,14)];

      vx(:,i,k) = v(:,1);
      vy(:,i,k) = v(:,2);
```

```
vz(:,i,k) = v(:,3);

v(1,:) = 0;
v(31,:) = 0;
v_p1(1,:) = 0;
v_p1(31,:) = 0;
v_m1(1,:) = 0;
v_m1(31,:) = 0;

xcoords(:,i,k) = polydata(point-30:point,1);
ycoords(:,i,k) = polydata(point-30:point,2);
zcoords(:,i,k) = polydata(point-30:point,3);

dvi_dxi(:,1,i) = polydata(point-30:point,7);
dvi_dxi(:,2,i) = polydata(point-30:point,8);
dvi_dxi(:,3,i) = polydata(point-30:point,9);
dvi_dxi(:,4,i) = polydata(point-30:point,11);
dvi_dxi(:,5,i) = polydata(point-30:point,12);
dvi_dxi(:,6,i) = polydata(point-30:point,13);
dvi_dxi(:,7,i) = polydata(point-30:point,15);
dvi_dxi(:,8,i) = polydata(point-30:point,16);
dvi_dxi(:,9,i) = polydata(point-30:point,17);

fluid_frac(:,i,k) = polydata(point-30:point,4);

%=========================================================================%
%                  Cross - Section Integration                  %
%=========================================================================%
% Here we integrate through each cross-section of data outputted by
% POLYFLOW. j = 1:31 because we have chosen 31 cross-section
% sampling points but the last sample is out of the domain
% resulting in NAN's so we have neglected that.
%=========================================================================%

% Calculating the orientation values at different cross-sections
% (j = 1 to # of cross-sections).

for j = 1:31 % Number of cross sections

   % Inital orientation guesses, starting at position 2 (which is
   % really equal to 1% fill). Further iterations in the i
   % direction use previous solutions of orientation.

   if (fluid_frac(j,i,k) < 0.15)
      continue
   end

   if (i == 2)
      [Br1 Br2 Br3] = BR_initial(j);
      Abr(j,:,1,k) = Br1;
      Bbr(j,:,1,k) = Br2;
      Cbr(j,:,1,k) = Br3;
   end

   % To estimate some of the derivatives in the models, we need to
   % calculate the change in coordinates. This calculates that
   % change as we interate through (j,i) steps (thickness,fill).
```

```
        dX = [0.2984 0.0667 0];

        % Reading the velocity gradient data at each node of interest
        % and manipulating the data to calculate the vorticity and rate
        % of deformation.
        dvi = [dvi_dxi(j,1,i) dvi_dxi(j,2,i) dvi_dxi(j,3,i) dvi_dxi(j,4,i) dvi_dxi(j,5,i) dvi_dxi(j,6,i) dvi_dxi(j,7,i) dvi_dxi(j,8,i)
dvi_dxi(j,9,i)];
        delV = reshape(dvi,3,3);
        w = reshape((1/2)*(delV - delV'),1,9);
        d = reshape((1/2)*(delV + delV'),1,9);

        % Defining the Input set of Orientation equation

        if ((j == 1) || (j == 31))
           A_0 = [Abr(j,:,i,k-step) Bbr(j,:,i,k-step) Cbr(j,:,i,k-step)];

           if (A_0(1) == 0)
              A_0 = [Abr(j,:,i-1,k) Bbr(j,:,i-1,k) Cbr(j,:,i-1,k)];
           end

           ftibof = @(t,A) BR_IBOF_mex(dX,dvi,v,v_m1,v_p1,w,d,lb,kbend,C_I,OF,A,i,j);
           [t1,A1] = ode45(ftibof,linspace(timesteps(k-step),timesteps(k),15),A_0);
           Abr(j,:,i,k) = A1(length(A1(:,1)),1:9);
           Bbr(j,:,i,k) = A1(length(A1(:,1)),10:18);
           Cbr(j,:,i,k) = A1(length(A1(:,1)),19:21);
        end

        if ((1 < j) && (j < 31))
           A_0 = [Abr(j,:,i,k-step) Bbr(j,:,i,k-step) Cbr(j,:,i,k-step)];

           if (A_0(1) == 0)
              A_0 = [Abr(j,:,i-1,k) Bbr(j,:,i-1,k) Cbr(j,:,i-1,k)];
           end

           Am11 = Abr(j,:,i-1,k);
           Am12 = Abr(j-1,:,i,k);
           Am13 = zeros(1,9);
           Bm11 = Bbr(j,:,i-1,k);
           Bm12 = Bbr(j-1,:,i,k);
           Bm13 = zeros(1,9);
           Cm11 = Cbr(j,:,i-1,k);
           Cm12 = Cbr(j-1,:,i,k);
           Cm13 = zeros(1,3);

           ftibof = @(t,A) BR_IBOFc_mex(dX,dvi,v,v_m1,v_p1,[vx(j,i,k) vy(j,i,k)
vz(j,i,k)],w,d,lb,kbend,C_I,OF,A,i,j,Am11,Am12,Am13,Bm11,Bm12,Bm13,Cm11,Cm12,Cm13);
           [t1,A1] = ode45(ftibof,linspace(timesteps(k-step),timesteps(k),15),A_0);
           Abr(j,:,i,k) = A1(length(A1(:,1)),1:9);
           Bbr(j,:,i,k) = A1(length(A1(:,1)),10:18);
           Cbr(j,:,i,k) = A1(length(A1(:,1)),19:21);
        end

        trB = Bbr(j,1,i,k) + Bbr(j,5,i,k) + Bbr(j,9,i,k);

        Rbr(j,1,i,k) = abs(Abr(j,1,i,k) - Bbr(j,1,i,k))./(1 - trB);
        Rbr(j,5,i,k) = abs(Abr(j,5,i,k) - Bbr(j,5,i,k))./(1 - trB);
        Rbr(j,9,i,k) = abs(Abr(j,9,i,k) - Bbr(j,9,i,k))./(1 - trB);
        Rbr(j,2,i,k) = abs(Abr(j,2,i,k) - Bbr(j,2,i,k))./(1 - trB);
```

```
        Rbr(j,3,i,k) = abs(Abr(j,3,i,k) - Abr(j,3,i,k))./(1 - trB);
      end

    end

    [x,y] = meshgrid(linspace(0,1,299),linspace(-1,1,31));
    figure(1)
    [C,h] = contourf(x,y,squeeze(Rbr(:,1,:,k)));
    clabel(C,h);
    set(gcf,'color','w','Position',[100 100 1200 300]);
    pause(0.01)
end
```

E.1.3 egp_ft.m

```
% K.J. Meyer
% 12/10/2011

clear all
close all
clc
savedir = 'D:\Computational Files\Transient EGP Simulation Files\Full Mold\EGP Mold\Experimental Full\FT Results Fitted
LinearInterp\';

%% Defining the POLYFLOW Files Directory
dir = 'D:\Computational Files\Transient EGP Simulation Files\Full Mold\EGP Mold\';
type = '.csv';
timesteps = dlmread(sprintf('%s%s%s',dir,'timesteps','.txt'),',');

%% Defining the Constants for the Simulation
OF = 0.0087;
C_I = 0.2278;

% b1 = 0.0018;
% b2 = 0.0005;
% b3 = 0.0013;
% b4 = 3.4E-5;
% b5 = 0.0015;

start = 40;
finish = 550;
fill = 299;
planes = 11;
step = 2;

%% Pre-allocation of internal variables for speed
vx = zeros(31,fill);
vy = zeros(31,fill);
vz = zeros(31,fill);

fluid_frac = zeros(31,fill,planes);

Aft_t2 = zeros(31,9,fill,planes);
Aft_t1 = zeros(31,9,fill,planes);

count = start;

%% Begin the Simulation
```

```
while (count <= finish) % Number of timesteps
   disp(count)
   data = dlmread(sprintf('%s%i%s',dir,count,type),',');
   polydata = sortrows(data, [3 1 2]);

   for k = 1:11

      plane_i_data = polydata(1 + 9300*(k-1):(9300*k),:);

      for i = 2:299
         % Number of fill percentages. Subtracting first and last step.
         % Numerical approximations are used and need i+1 and i-1 data.
         % These are not availible for 1 and 100.
         %=========================================================================%
         % To calculate the change in distance in the flow direction in the BR
         % model we need to load the cross-section data from the previous
         % two positions. Special conditions are needed for the first two
         % iterative steps in the solution because i /= 0;
         %=========================================================================%



         %=========================================================================%
         % Reading in the data for the cross-section of interest. For some
         % of the calculation we need one and two steps back from the region
         % of interest to estimate the derivative.
         %=========================================================================%

         clear dvi_dxi

         pointm1 = (i-1)*31;
         point = i*31;
         pointp1 = (i+1)*31;

         v = [plane_i_data(point-30:point,6) plane_i_data(point-30:point,10) plane_i_data(point-30:point,14)];
         v_m1 = [plane_i_data(pointm1-30:pointm1,6) plane_i_data(pointm1-30:pointm1,10) plane_i_data(pointm1-
30:pointm1,14)];
         v_p1 = [plane_i_data(pointp1-30:pointp1,6) plane_i_data(pointp1-30:pointp1,10) plane_i_data(pointp1-30:pointp1,14)];

         % For visualization purposes

         vx(:,i,k) = v(:,1);
         vy(:,i,k) = v(:,2);
         vz(:,i,k) = v(:,3);

         dvi_dxi = zeros(31,9,fill);

         dvi_dxi(:,1,i) = plane_i_data(point-30:point,7);
         dvi_dxi(:,2,i) = plane_i_data(point-30:point,8);
         dvi_dxi(:,3,i) = plane_i_data(point-30:point,9);
         dvi_dxi(:,4,i) = plane_i_data(point-30:point,11);
         dvi_dxi(:,5,i) = plane_i_data(point-30:point,12);
         dvi_dxi(:,6,i) = plane_i_data(point-30:point,13);
         dvi_dxi(:,7,i) = plane_i_data(point-30:point,15);
         dvi_dxi(:,8,i) = plane_i_data(point-30:point,16);
         dvi_dxi(:,9,i) = plane_i_data(point-30:point,17);

         fluid_frac(:,i,k) = plane_i_data(point-30:point,4);
```

```
%% ========================================================================%
%                    Cross - Section Integration                    %
%=========================================================================%
% Here we integrate through each cross-section of data outputted by
% POLYFLOW. j = 1:31 because we have chosen 31 cross-section
% sampling points but the last sample is out of the domain
% resulting in NAN's so we have neglected that.
%=========================================================================%

% Calculating the orientation values at different cross-sections
% (j = 1 to # of cross-sections).

for j = 1:31 % Number of cross sections

   % Inital orientation guesses, starting at position 2 (which is
   % really equal to 1% fill). Further iterations in the i
   % direction use previous solutions of orientation.

   if (fluid_frac(j,i,k) < 0.15)
      continue
   end

   if (i == 2)
      Aft_t2(j,:,1,k) = EXP_EGP_full(i,j,k,count);
      Aft_t2(j,:,i,k) = EXP_EGP_full(i,j,k,count);
   end

   % To estimate some of the derivatives in the models, we need to
   % calculate the change in coordinates. This calculates that
   % change as we interate through (j,i) steps (thickness,fill).

   dX = [0.4 0.0500 0];

   if (k > 1)
      dX(1,3) = 3.4114;
   end

   % Reading the velocity gradient data at each node of interest
   % and manipulating the data to calculate the vorticity and rate
   % of deformation.
   dvi = [dvi_dxi(j,1,i) dvi_dxi(j,2,i) dvi_dxi(j,3,i) dvi_dxi(j,4,i) dvi_dxi(j,5,i) dvi_dxi(j,6,i) dvi_dxi(j,7,i) dvi_dxi(j,8,i)
dvi_dxi(j,9,i)];
   delV = reshape(dvi,3,3);
   w = reshape((1/2)*(delV - delV'),1,9);
   d = reshape((1/2)*(delV + delV'),1,9);

   % Defining the Input set of Orientation equations

   if ((j == 1) || (j == 31))
      A_0 = Aft_t1(j,:,i,k);

      if (A_0(1) == 0)
         A_0 = Aft_t2(j,:,i-1,k);
      end

%        ftibof = @(t,A) ARD_RSC_IBOF_mex(w,d,OF,b1,b2,b3,b4,b5,A);
```

```
            ftibof = @(t,A) FT_IBOF_mex(w,d,C_I,OF,A);
            [t1,A1] = ode45(ftibof,linspace(timesteps(count-step),timesteps(count),5),A_0);
            Aft_t2(j,:,i,k) = A1(length(A1(:,1)),:);
            Aft_t2(j,:,i,k) = Aft_t2(j,:,i,k)./(Aft_t2(j,1,i,k) + Aft_t2(j,5,i,k) + Aft_t2(j,9,i,k));
            continue
        end

        if ((1 < j) && (j < 31))
            A_0 = Aft_t1(j,:,i,k);

            if (A_0(1) == 0)
                A_0 = Aft_t2(j,:,i-1,k);
            end

            Am11 = Aft_t2(j,:,i-1,k);
            Am12 = Aft_t2(j-1,:,i,k);
            Am13 = zeros(1,9);

            if (k > 1)
                Am13 = Aft_t2(j,:,i,k-1);
                A_0 = Aft_t2(j,:,i,k-1);
            end
%               ftibof = @(t,A) ARD_RSC_IBOFc_mex(w,d,OF,b1,b2,b3,b4,b5,A,Am11,Am12,Am13,dX,[vx(j,i,k) vy(j,i,k)
vz(j,i,k)]);
            ftibof = @(t,A) FT_IBOFc_mex(w,d,C_I,OF,A,Am11,Am12,Am13,dX,[vx(j,i,k) vy(j,i,k) vz(j,i,k)]);
            [t1,A1] = ode45(ftibof,linspace(timesteps(count-step),timesteps(count),5),A_0);
            Aft_t2(j,:,i,k) = A1(length(A1(:,1)),:);
            Aft_t2(j,:,i,k) = Aft_t2(j,:,i,k)./(Aft_t2(j,1,i,k) + Aft_t2(j,5,i,k) + Aft_t2(j,9,i,k));
        end
    end
  end
  if (count > 5)
    dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'A11ft_cross',k,'_',count,'.txt'),squeeze(Aft_t2(:,1,:,k)));
    dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'A12ft_cross',k,'_',count,'.txt'),squeeze(Aft_t2(:,2,:,k)));
    dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'A13ft_cross',k,'_',count,'.txt'),squeeze(Aft_t2(:,3,:,k)));
    dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'A22ft_cross',k,'_',count,'.txt'),squeeze(Aft_t2(:,5,:,k)));
    dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'A33ft_cross',k,'_',count,'.txt'),squeeze(Aft_t2(:,9,:,k)));
  end
 end
 count = count + step;
 Aft_t1 = Aft_t2;
end
```

E.1.4 egp_br.m

```
%% Transient Cross-Section Orientation Calculations
% K.J. Meyer
% 12/10/2011

clear all
close all
clc
savedir = 'D:\Computational Files\Transient EGP Simulation Files\Full Mold\EGP Mold\Experimental\BR Results Fitted\';

%% Defining the POLYFLOW Files Directory
dir = 'D:\Computational Files\Transient EGP Simulation Files\Full Mold\EGP Mold\';
type = '.csv';
timesteps = dlmread(sprintf('%s%s%s',dir,'timesteps','.txt'),',');
```

```
% Loading A Previous Solution
load = 1;

%% Defining the Constants for the Simulation
OF = 0.0039;
C_I = 0.4843;
lb = 1.9;       % mm
dia = 0.1;      % mm
eta_0 = 293;   % Pa*s
lam = 0.0059;   % s^-1
n = 0.24;     % none
a = 0.749;
Ey = 80E9;      % Pa

start = 40;
finish = 550;
fill = 299;
planes = 11;
step = 2;

%% Pre-allocation of internal variables for speed
vx = zeros(31,fill);
vy = zeros(31,fill);
vz = zeros(31,fill);

fluid_frac = zeros(31,fill,planes);

Abr_t2 = zeros(31,9,fill,planes);
Abr_t1 = zeros(31,9,fill,planes);
Bbr_t2 = zeros(31,9,fill,planes);
Bbr_t1 = zeros(31,9,fill,planes);
Cbr_t2 = zeros(31,3,fill,planes);
Cbr_t1 = zeros(31,3,fill,planes);

count = start;

%% Begin the Simulation
while (count <= finish) % Number of timesteps
   disp(count)
   data = dlmread(sprintf('%s%i%s',dir,count,'.csv'));
   polydata = sortrows(data, [3 1 2]);

   for k = 1

      plane_i_data = polydata(1 + 9300*(k-1):(9300*k),:);

      for i = 2:299
         % Number of fill percentages. Subtracting first and last step.
         % Numerical approximations are used and need i+1 and i-1 data.
         % These are not availible for 1 and 300
         %=========================================================================%
         % To calculate the change in distance in the flow direction in the BR
         % model we need to load the cross-section data from the previous
         % two positions. Special conditions are needed for the first two
         % iterative steps in the solution because i /= 0;
         %=========================================================================%
```

```
%=========================================================================%
% Reading in the data for the cross-section of interest. For some
% of the calculation we need one and two steps back from the region
% of interest to estimate the derivative.
%=========================================================================%

pointm1 = (i-1)*31;
point = i*31;
pointp1 = (i+1)*31;

v = [plane_i_data(point-30:point,6) plane_i_data(point-30:point,10) plane_i_data(point-30:point,14)];
v_m1 = [plane_i_data(pointm1-30:pointm1,6) plane_i_data(pointm1-30:pointm1,10) plane_i_data(pointm1-
30:pointm1,14)];
v_p1 = [plane_i_data(pointp1-30:pointp1,6) plane_i_data(pointp1-30:pointp1,10) plane_i_data(pointp1-30:pointp1,14)];

% For visualization purposes

vx(:,i) = v(:,1);
vy(:,i) = v(:,2);
vz(:,i) = v(:,3);

dvi_dxi = zeros(31,9,fill);

dvi_dxi(:,1,i) = plane_i_data(point-30:point,7);
dvi_dxi(:,2,i) = plane_i_data(point-30:point,8);
dvi_dxi(:,3,i) = plane_i_data(point-30:point,9);
dvi_dxi(:,4,i) = plane_i_data(point-30:point,11);
dvi_dxi(:,5,i) = plane_i_data(point-30:point,12);
dvi_dxi(:,6,i) = plane_i_data(point-30:point,13);
dvi_dxi(:,7,i) = plane_i_data(point-30:point,15);
dvi_dxi(:,8,i) = plane_i_data(point-30:point,16);
dvi_dxi(:,9,i) = plane_i_data(point-30:point,17);

fluid_frac(:,i,k) = plane_i_data(point-30:point,4);

%% =========================================================================%
%                  Cross - Section Integration                  %
%=========================================================================%
% Here we integrate through each cross-section of data outputted by
% POLYFLOW. j = 1:31 because we have chosen 31 cross-section
% sampling points but the last sample is out of the domain
% resulting in NAN's so we have neglected that.
%=========================================================================%

% Calculating the orientation values at different cross-sections
% (j = 1 to # of cross-sections).

for j = 1:31 % Number of cross sections

   if (fluid_frac(j,i,k) < 0.15)
      continue
   end

   if (i == 2)
      Abr_t2(j,:,1,k) = EXP_EGP_full(i,j,k,count);
      Bbr_t2(j,:,1,k) = -EXP_EGP_full(i,j,k,count);
      Cbr_t2(j,:,1,k) = [0 0 0];
      Abr_t2(j,:,i,k) = EXP_EGP_full(i,j,k,count);
```

```
            Bbr_t2(j,:,i,k) = -EXP_EGP_full(i,j,k,count);
            Cbr_t2(j,:,i,k) = [0 0 0];
        end

        % To estimate some of the derivatives in the models, we need to
        % calculate the change in coordinates. This calculates that
        % change as we interate through (j,i) steps (thickness,fill).

        dX = [0.4 0.0500 0];

        if (k > 1)
            dX(1,3) = 3.4114;
        end

        % Reading the velocity gradient data at each node of interest
        % and manipulating the data to calculate the vorticity and rate
        % of deformation.
        dvi = [dvi_dxi(j,1,i) dvi_dxi(j,2,i) dvi_dxi(j,3,i) dvi_dxi(j,4,i) dvi_dxi(j,5,i) dvi_dxi(j,6,i) dvi_dxi(j,7,i) dvi_dxi(j,8,i)
dvi_dxi(j,9,i)];
        delV = reshape(dvi,3,3);
        w = reshape((1/2)*(delV - delV'),1,9);
        d = reshape((1/2)*(delV + delV'),1,9);

        gamma = 2*d;
        gamma_mag = sqrt((1/2)*(gamma(1)*gamma(1) + gamma(2)*gamma(2) + gamma(3)*gamma(3) + ...
                        gamma(4)*gamma(4) + gamma(5)*gamma(5) + gamma(6)*gamma(6) + ...
                        gamma(7)*gamma(7) + gamma(8)*gamma(8) + gamma(9)*gamma(9)));

        if ((j == 1) || (j == 31))
            A0 = [Abr_t1(j,:,i,k) Bbr_t1(j,:,i,k) Cbr_t1(j,:,i,k)];

            if (A0(1) == 0)
                A0 = [Abr_t2(j,:,i-1,k) Bbr_t2(j,:,i-1,k) Cbr_t2(j,:,i-1,k)];
            end

%               eta_m = eta_0*(1+(lam*gamma_mag)^a)^((n-1)/a);
            kflex = 20; %(Ey/(64*eta_m))*(dia^3/lb^3);

            bribof = @(t,A) BR_IBOF_mex(dX,dvi,v,v_m1,v_p1,w,d,lb,kflex,C_I,OF,A,i,j);
            [t1,A1] = ode45(bribof,linspace(timesteps(count-step),timesteps(count),10),A0);
            Abr_t2(j,:,i,k) = A1(length(A1(:,1)),1:9);
            Bbr_t2(j,:,i,k) = A1(length(A1(:,1)),10:18);
            Cbr_t2(j,:,i,k) = A1(length(A1(:,1)),19:21);
            continue
        end

        % Defining the Input set of Orientation equations

        if ((1 < j) && (j < 31))

            A0 = [Abr_t1(j,:,i,k) Bbr_t1(j,:,i,k) Cbr_t1(j,:,i,k)];

            if (A0(1) == 0)
                A0 = [Abr_t2(j,:,i-1,k) Bbr_t2(j,:,i-1,k) Cbr_t2(j,:,i-1,k)];
            end

            Am11 = Abr_t2(j,:,i-1,k);
```

```
            Bm11 = Bbr_t2(j,:,i-1,k);
            Cm11 = Cbr_t2(j,:,i-1,k);
            Am12 = Abr_t2(j-1,:,i,k);
            Bm12 = Bbr_t2(j-1,:,i,k);
            Cm12 = Cbr_t2(j-1,:,i,k);
            Am13 = zeros(1,9);
            Bm13 = zeros(1,9);
            Cm13 = zeros(1,3);

            if (k > 1)
               Am13 = Abr_t2(j,:,i,k-1);
               Bm13 = Bbr_t2(j,:,i,k-1);
               Cm13 = Cbr_t2(j,:,i,k-1);
%                 A0 = [Abr_t2(j,:,i,k-1) Bbr_t2(j,:,i,k-1) Cbr_t2(j,:,i,k-1)];
            end

%             eta_m = eta_0*(1+(lam*gamma_mag)^a)^((n-1)/a);
            kflex = 20; %(Ey/(64*eta_m))*(dia^3/lb^3);

            bribof = @(t,A) BR_IBOFc_mex(dX,dvi,v,v_m1,v_p1,[vx(j,i) vy(j,i)
vz(j,i)],w,d,lb,kflex,C_I,OF,A,i,j,Am11,Am12,Am13,Bm11,Bm12,Bm13,Cm11,Cm12,Cm13);
            [t1,A1] = ode45(bribof,linspace(timesteps(count-step),timesteps(count),10),A0);
            Abr_t2(j,:,i,k) = A1(length(A1(:,1)),1:9);
            Bbr_t2(j,:,i,k) = A1(length(A1(:,1)),10:18);
            Cbr_t2(j,:,i,k) = A1(length(A1(:,1)),19:21);
         end

         if (Abr_t2(j,1,i,k) >= 1) || (Abr_t2(j,5,i,k) >= 1) || (Abr_t2(j,9,i,k) >= 1) || (Abr_t2(j,1,i,k) <= 0) || (Abr_t2(j,5,i,k) <= 0)
|| (Abr_t2(j,9,i,k) <= 0) || isnan(Abr_t2(j,1,i,k))
            [Ai Bi Ci Ri] = threed_br_error_func(i,j,k,2,dX,dvi,v,v_m1,v_p1,w,d,lb,kflex,C_I,OF,[timesteps(count-step)
timesteps(count)]);
            Abr_t2(j,:,i,k) = Ai;
            Bbr_t2(j,:,i,k) = Bi;
            Cbr_t2(j,:,i,k) = Ci;
         end
      end
   end
   if (count > 5)
      dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'A11br_cross',k,'_',count,'.txt'),squeeze(Abr_t2(:,1,:,k)));
      dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'A12br_cross',k,'_',count,'.txt'),squeeze(Abr_t2(:,2,:,k)));
      dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'A13br_cross',k,'_',count,'.txt'),squeeze(Abr_t2(:,3,:,k)));
      dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'A22br_cross',k,'_',count,'.txt'),squeeze(Abr_t2(:,5,:,k)));
      dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'A33br_cross',k,'_',count,'.txt'),squeeze(Abr_t2(:,9,:,k)));
      dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'B11br_cross',k,'_',count,'.txt'),squeeze(Bbr_t2(:,1,:,k)));
      dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'B12br_cross',k,'_',count,'.txt'),squeeze(Bbr_t2(:,2,:,k)));
      dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'B13br_cross',k,'_',count,'.txt'),squeeze(Bbr_t2(:,3,:,k)));
      dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'B22br_cross',k,'_',count,'.txt'),squeeze(Bbr_t2(:,5,:,k)));
      dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'B33br_cross',k,'_',count,'.txt'),squeeze(Bbr_t2(:,9,:,k)));
      dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'C11br_cross',k,'_',count,'.txt'),squeeze(Cbr_t2(:,1,:,k)));
      dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'C22br_cross',k,'_',count,'.txt'),squeeze(Cbr_t2(:,2,:,k)));
      dlmwrite(sprintf('%s%s%i%s%i%s',savedir,'C33br_cross',k,'_',count,'.txt'),squeeze(Cbr_t2(:,3,:,k)));
   end
end
count = count + step;
Abr_t1 = Abr_t2;
Bbr_t1 = Bbr_t2;
Cbr_t1 = Cbr_t2;
end
```

E.2 Core Routines

E.2.1 FT_quad.m

```
function [dA] = FT_quad(w,d,C_I,OF,A)
%% Parameter Definitions

W = w;
D = d;
gamma = 2*D;
I = [1 0 0 0 1 0 0 0 1];

%% Double-Dot Terms
gamma_mag = sqrt((1/2)*(gamma(1)*gamma(1) + gamma(2)*gamma(4) + gamma(3)*gamma(7) + ...
                    gamma(4)*gamma(2) + gamma(5)*gamma(5) + gamma(6)*gamma(8) + ...
                    gamma(7)*gamma(3) + gamma(8)*gamma(6) + gamma(9)*gamma(9)));


%% Orientation Calculations

dA = zeros(9,1);

dA(1) = OF*(((W(1)*A(1) + W(2)*A(4) + W(3)*A(7)) - (A(1)*W(1) + A(2)*W(4) + A(3)*W(7))) + ...
        ((D(1)*A(1) + D(2)*A(4) + D(3)*A(7)) + (A(1)*D(1) + A(2)*D(4) + A(3)*D(7)) - ...
        2*Explicit_Quad_v(D,A)*A(1)) + 2 * C_I * gamma_mag * (I(1) - 3 * A(1)));

dA(2) = OF*(((W(1)*A(2) + W(2)*A(5) + W(3)*A(8)) - (A(1)*W(2) + A(2)*W(5) + A(3)*W(8))) + ...
        ((D(1)*A(2) + D(2)*A(5) + D(3)*A(8)) + (A(1)*D(2) + A(2)*D(5) + A(3)*D(8)) - ...
        2*Explicit_Quad_v(D,A)*A(2)) + 2 * C_I * gamma_mag * (I(2) - 3 * A(2)));

dA(3) = OF*(((W(1)*A(3) + W(2)*A(6) + W(3)*A(9)) - (A(1)*W(3) + A(2)*W(6) + A(3)*W(9))) + ...
        ((D(1)*A(3) + D(2)*A(6) + D(3)*A(9)) + (A(1)*D(3) + A(2)*D(6) + A(3)*D(9)) - ...
        2*Explicit_Quad_v(D,A)*A(3)) + 2 * C_I * gamma_mag * (I(3) - 3 * A(3)));

dA(4) = dA(2);

dA(5) = OF*(((W(4)*A(2) + W(5)*A(5) + W(6)*A(8)) - (A(4)*W(2) + A(5)*W(5) + A(6)*W(8))) + ...
        ((D(4)*A(2) + D(5)*A(5) + D(6)*A(8)) + (A(4)*D(2) + A(5)*D(5) + A(6)*D(8)) - ...
        2*Explicit_Quad_v(D,A)*A(5)) + 2 * C_I * gamma_mag * (I(5) - 3 * A(5)));

dA(6) = OF*(((W(4)*A(3) + W(5)*A(6) + W(6)*A(9)) - (A(4)*W(3) + A(5)*W(6) + A(6)*W(9))) + ...
        ((D(4)*A(3) + D(5)*A(6) + D(6)*A(9)) + (A(4)*D(3) + A(5)*D(6) + A(6)*D(9)) - ...
        2*Explicit_Quad_v(D,A)*A(6)) + 2 * C_I * gamma_mag * (I(6) - 3 * A(6)));

dA(7) = dA(3);

dA(8) = dA(6);

dA(9) = OF*(((W(7)*A(3) + W(8)*A(6) + W(9)*A(9)) - (A(7)*W(3) + A(8)*W(6) + A(9)*W(9))) + ...
        ((D(7)*A(3) + D(8)*A(6) + D(9)*A(9)) + (A(7)*D(3) + A(8)*D(6) + A(9)*D(9)) - ...
        2*Explicit_Quad_v(D,A)*A(9)) + 2 * C_I * gamma_mag * (I(9) - 3 * A(9)));


end
```

E.2.2 FT_quadc.m

```
function [dA] = FT_quadc(w,d,C_I,OF,A,Am11,Am12,Am13,dX,v)
%% Parameter Definitions

W = w;
D = d;
gamma = 2*D;
I = [1 0 0 0 1 0 0 0 1];

%% Double-Dot Terms
gamma_mag = sqrt((1/2)*(gamma(1)*gamma(1) + gamma(2)*gamma(4) + gamma(3)*gamma(7) + ...
                    gamma(4)*gamma(2) + gamma(5)*gamma(5) + gamma(6)*gamma(8) + ...
                    gamma(7)*gamma(3) + gamma(8)*gamma(6) + gamma(9)*gamma(9)));

%% Orientation Calculations

convA = convective(A,Am11,Am12,Am13,v,dX);

dA = zeros(9,1);

dA(1) = OF*(((W(1)*A(1) + W(2)*A(4) + W(3)*A(7)) - (A(1)*W(1) + A(2)*W(4) + A(3)*W(7))) + ...
        ((D(1)*A(1) + D(2)*A(4) + D(3)*A(7)) + (A(1)*D(1) + A(2)*D(4) + A(3)*D(7)) - ...
        2*Explicit_Quad_v(D,A)*A(1)) + 2 * C_I * gamma_mag * (I(1) - 3 * A(1))) - convA(1);

dA(2) = OF*(((W(1)*A(2) + W(2)*A(5) + W(3)*A(8)) - (A(1)*W(2) + A(2)*W(5) + A(3)*W(8))) + ...
        ((D(1)*A(2) + D(2)*A(5) + D(3)*A(8)) + (A(1)*D(2) + A(2)*D(5) + A(3)*D(8)) - ...
        2*Explicit_Quad_v(D,A)*A(2)) + 2 * C_I * gamma_mag * (I(2) - 3 * A(2))) - convA(2);

dA(3) = OF*(((W(1)*A(3) + W(2)*A(6) + W(3)*A(9)) - (A(1)*W(3) + A(2)*W(6) + A(3)*W(9))) + ...
        ((D(1)*A(3) + D(2)*A(6) + D(3)*A(9)) + (A(1)*D(3) + A(2)*D(6) + A(3)*D(9)) - ...
        2*Explicit_Quad_v(D,A)*A(3)) + 2 * C_I * gamma_mag * (I(3) - 3 * A(3))) - convA(3);

dA(4) = dA(2);

dA(5) = OF*(((W(4)*A(2) + W(5)*A(5) + W(6)*A(8)) - (A(4)*W(2) + A(5)*W(5) + A(6)*W(8))) + ...
        ((D(4)*A(2) + D(5)*A(5) + D(6)*A(8)) + (A(4)*D(2) + A(5)*D(5) + A(6)*D(8)) - ...
        2*Explicit_Quad_v(D,A)*A(5)) + 2 * C_I * gamma_mag * (I(5) - 3 * A(5))) - convA(5);

dA(6) = OF*(((W(4)*A(3) + W(5)*A(6) + W(6)*A(9)) - (A(4)*W(3) + A(5)*W(6) + A(6)*W(9))) + ...
        ((D(4)*A(3) + D(5)*A(6) + D(6)*A(9)) + (A(4)*D(3) + A(5)*D(6) + A(6)*D(9)) - ...
        2*Explicit_Quad_v(D,A)*A(6)) + 2 * C_I * gamma_mag * (I(6) - 3 * A(6))) - convA(6);

dA(7) = dA(3);

dA(8) = dA(6);

dA(9) = OF*(((W(7)*A(3) + W(8)*A(6) + W(9)*A(9)) - (A(7)*W(3) + A(8)*W(6) + A(9)*W(9))) + ...
        ((D(7)*A(3) + D(8)*A(6) + D(9)*A(9)) + (A(7)*D(3) + A(8)*D(6) + A(9)*D(9)) - ...
        2*Explicit_Quad_v(D,A)*A(9)) + 2 * C_I * gamma_mag * (I(9) - 3 * A(9))) - convA(9);


end
```

E.2.3 FT_IBOF.m

```
function [dA] = FT_IBOFc(w,d,C_I,OF,A,Am11,Am12,Am13,dX,v)
%# codegen

%% Parameter Definitions
W = w;
D = d;
gamma = 2*D;
I = [1 0 0 0 1 0 0 0 1];

%% Double-Dot Terms
gamma_mag = gamma(1)*gamma(1) + gamma(2)*gamma(4) + gamma(3)*gamma(7) + ...
        gamma(4)*gamma(2) + gamma(5)*gamma(5) + gamma(6)*gamma(8) + ...
        gamma(7)*gamma(3) + gamma(8)*gamma(6) + gamma(9)*gamma(9);

%% Folgar-Tucker Equations

convA = convective(A,Am11,Am12,Am13,v,dX);

dA = zeros(9,1);

dA(1) = OF*(((W(1)*A(1) + W(2)*A(4) + W(3)*A(7)) - (A(1)*W(1) + A(2)*W(4) + A(3)*W(7))) + ...
   ((D(1)*A(1) + D(2)*A(4) + D(3)*A(7)) + (A(1)*D(1) + A(2)*D(4) + A(3)*D(7)) - ...
   2*Explicit_IBOF_v(D,A,1,1)) + 2 * C_I * sqrt((1/2)*gamma_mag) * (I(1) - 3 * A(1)));

dA(2) = OF*(((W(1)*A(2) + W(2)*A(5) + W(3)*A(8)) - (A(1)*W(2) + A(2)*W(5) + A(3)*W(8))) + ...
   ((D(1)*A(2) + D(2)*A(5) + D(3)*A(8)) + (A(1)*D(2) + A(2)*D(5) + A(3)*D(8)) - ...
   2*Explicit_IBOF_v(D,A,1,2)) + 2 * C_I * sqrt((1/2)*gamma_mag) * (I(2) - 3 * A(2)));

dA(3) = OF*(((W(1)*A(3) + W(2)*A(6) + W(3)*A(9)) - (A(1)*W(3) + A(2)*W(6) + A(3)*W(9))) + ...
   ((D(1)*A(3) + D(2)*A(6) + D(3)*A(9)) + (A(1)*D(3) + A(2)*D(6) + A(3)*D(9)) - ...
   2*Explicit_IBOF_v(D,A,1,3)) + 2 * C_I * sqrt((1/2)*gamma_mag) * (I(3) - 3 * A(3)));

dA(4) = dA(2);

dA(5) = OF*(((W(4)*A(2) + W(5)*A(5) + W(6)*A(8)) - (A(4)*W(2) + A(5)*W(5) + A(6)*W(8))) + ...
   ((D(4)*A(2) + D(5)*A(5) + D(6)*A(8)) + (A(4)*D(2) + A(5)*D(5) + A(6)*D(8)) - ...
   2*Explicit_IBOF_v(D,A,2,2)) + 2 * C_I * sqrt((1/2)*gamma_mag) * (I(5) - 3 * A(5)));

dA(6) = OF*(((W(4)*A(3) + W(5)*A(6) + W(6)*A(9)) - (A(4)*W(3) + A(5)*W(6) + A(6)*W(9))) + ...
   ((D(4)*A(3) + D(5)*A(6) + D(6)*A(9)) + (A(4)*D(3) + A(5)*D(6) + A(6)*D(9)) - ...
   2*Explicit_IBOF_v(D,A,2,3)) + 2 * C_I * sqrt((1/2)*gamma_mag) * (I(6) - 3 * A(6)));

dA(7) = dA(3);

dA(8) = dA(6);

dA(9) = OF*(((W(7)*A(3) + W(8)*A(6) + W(9)*A(9)) - (A(7)*W(3) + A(8)*W(6) + A(9)*W(9))) + ...
   ((D(7)*A(3) + D(8)*A(6) + D(9)*A(9)) + (A(7)*D(3) + A(8)*D(6) + A(9)*D(9)) - ...
   2*Explicit_IBOF_v(D,A,3,3)) + 2 * C_I * sqrt((1/2)*gamma_mag) * (I(9) - 3 * A(9)));

end
```

E.2.4 FT_IBOFc.m

```
function [dA] = FT_IBOFc(w,d,C_I,OF,A,Am11,Am12,Am13,dX,v)
%# codegen

%% Parameter Definitions
W = w;
D = d;
gamma = 2*D;
I = [1 0 0 0 1 0 0 0 1];

%% Double-Dot Terms
gamma_mag = gamma(1)*gamma(1) + gamma(2)*gamma(4) + gamma(3)*gamma(7) + ...
        gamma(4)*gamma(2) + gamma(5)*gamma(5) + gamma(6)*gamma(8) + ...
        gamma(7)*gamma(3) + gamma(8)*gamma(6) + gamma(9)*gamma(9);

%% Folgar-Tucker Equations

convA = convective(A,Am11,Am12,Am13,v,dX);

dA = zeros(9,1);

dA(1) = OF*(((W(1)*A(1) + W(2)*A(4) + W(3)*A(7)) - (A(1)*W(1) + A(2)*W(4) + A(3)*W(7))) + ...
   ((D(1)*A(1) + D(2)*A(4) + D(3)*A(7)) + (A(1)*D(1) + A(2)*D(4) + A(3)*D(7)) - ...
   2*Explicit_IBOF_v(D,A,1,1)) + 2 * C_I * sqrt((1/2)*gamma_mag) * (I(1) - 3 * A(1))) - convA(1);

dA(2) = OF*(((W(1)*A(2) + W(2)*A(5) + W(3)*A(8)) - (A(1)*W(2) + A(2)*W(5) + A(3)*W(8))) + ...
   ((D(1)*A(2) + D(2)*A(5) + D(3)*A(8)) + (A(1)*D(2) + A(2)*D(5) + A(3)*D(8)) - ...
   2*Explicit_IBOF_v(D,A,1,2)) + 2 * C_I * sqrt((1/2)*gamma_mag) * (I(2) - 3 * A(2))) - convA(2);

dA(3) = OF*(((W(1)*A(3) + W(2)*A(6) + W(3)*A(9)) - (A(1)*W(3) + A(2)*W(6) + A(3)*W(9))) + ...
   ((D(1)*A(3) + D(2)*A(6) + D(3)*A(9)) + (A(1)*D(3) + A(2)*D(6) + A(3)*D(9)) - ...
   2*Explicit_IBOF_v(D,A,1,3)) + 2 * C_I * sqrt((1/2)*gamma_mag) * (I(3) - 3 * A(3))) - convA(3);

dA(4) = dA(2);

dA(5) = OF*(((W(4)*A(2) + W(5)*A(5) + W(6)*A(8)) - (A(4)*W(2) + A(5)*W(5) + A(6)*W(8))) + ...
   ((D(4)*A(2) + D(5)*A(5) + D(6)*A(8)) + (A(4)*D(2) + A(5)*D(5) + A(6)*D(8)) - ...
   2*Explicit_IBOF_v(D,A,2,2)) + 2 * C_I * sqrt((1/2)*gamma_mag) * (I(5) - 3 * A(5))) - convA(5);

dA(6) = OF*(((W(4)*A(3) + W(5)*A(6) + W(6)*A(9)) - (A(4)*W(3) + A(5)*W(6) + A(6)*W(9))) + ...
   ((D(4)*A(3) + D(5)*A(6) + D(6)*A(9)) + (A(4)*D(3) + A(5)*D(6) + A(6)*D(9)) - ...
   2*Explicit_IBOF_v(D,A,2,3)) + 2 * C_I * sqrt((1/2)*gamma_mag) * (I(6) - 3 * A(6))) - convA(6);

dA(7) = dA(3);

dA(8) = dA(6);

dA(9) = OF*(((W(7)*A(3) + W(8)*A(6) + W(9)*A(9)) - (A(7)*W(3) + A(8)*W(6) + A(9)*W(9))) + ...
   ((D(7)*A(3) + D(8)*A(6) + D(9)*A(9)) + (A(7)*D(3) + A(8)*D(6) + A(9)*D(9)) - ...
   2*Explicit_IBOF_v(D,A,3,3)) + 2 * C_I * sqrt((1/2)*gamma_mag) * (I(9) - 3 * A(9))) - convA(9);

end
```

E.2.5 BR_IBOF.m

```
function [dA] = BR_IBOF(dX,dvi,v,v_m1,v_p1,w,d,lb,kk,C_I,OF,A,i,j)
%% Parameter Definitions

W = w;
D = d;
gamma = 2*D;
I = [1 0 0 0 1 0 0 0 1];

%% Double-Dot Terms
gamma_mag = sqrt((1/2)*(gamma(1)*gamma(1) + gamma(2)*gamma(2) + gamma(3)*gamma(3) + ...
                    gamma(4)*gamma(4) + gamma(5)*gamma(5) + gamma(6)*gamma(6) + ...
                    gamma(7)*gamma(7) + gamma(8)*gamma(8) + gamma(9)*gamma(9)));

%% Calculation of trace
trB = A(10) + A(14) + A(18);

%% Calculation of m*C
mdotc = mvector(dX,v,v_m1,v_p1,A,1,i,j)*A(19) + mvector(dX,v,v_m1,v_p1,A,2,i,j)*A(20) +
mvector(dX,v,v_m1,v_p1,A,3,i,j)*A(21);

%% Calculation of D:A
ddoublea = 2*(D(1)*A(1) + D(2)*A(4) + D(3)*A(7) + ...
          D(4)*A(2) + D(5)*A(5) + D(6)*A(8) + ...
          D(7)*A(3) + D(8)*A(6) + D(9)*A(9));

%% A Orientation Tensor Calculations

dA = zeros(21,1);

dA(1) = OF*(((W(1)*A(1) + W(2)*A(4) + W(3)*A(7)) - (A(1)*W(1) + A(2)*W(4) + A(3)*W(7))) + ((D(1)*A(1) + D(2)*A(4) +
D(3)*A(7)) + (A(1)*D(1) + A(2)*D(4) + A(3)*D(7)) - 2*Explicit_IBOF_v(D,A,1,1)) + ...
          (lb/2)*(A(19)*mvector(dX,v,v_m1,v_p1,A,1,i,j) + mvector(dX,v,v_m1,v_p1,A,1,i,j)*A(19) - 2*(mdotc)*A(1)) -
2*kk*(A(10) - A(1)*trB) - 6*C_I * gamma_mag * (A(1) - (1/3) * I(1)));

dA(2) = OF*(((W(1)*A(2) + W(2)*A(5) + W(3)*A(8)) - (A(1)*W(2) + A(2)*W(5) + A(3)*W(8))) + ((D(1)*A(2) + D(2)*A(5) +
D(3)*A(8)) + (A(1)*D(2) + A(2)*D(5) + A(3)*D(8)) - 2*Explicit_IBOF_v(D,A,1,2)) + ...
          (lb/2)*(A(19)*mvector(dX,v,v_m1,v_p1,A,2,i,j) + mvector(dX,v,v_m1,v_p1,A,1,i,j)*A(20) - 2*(mdotc)*A(2)) -
2*kk*(A(11) - A(2)*trB) - 6*C_I * gamma_mag * (A(2) - (1/3) * I(2)));

dA(3) = OF*(((W(1)*A(3) + W(2)*A(6) + W(3)*A(9)) - (A(1)*W(3) + A(2)*W(6) + A(3)*W(9))) + ((D(1)*A(3) + D(2)*A(6) +
D(3)*A(9)) + (A(1)*D(3) + A(2)*D(6) + A(3)*D(9)) - 2*Explicit_IBOF_v(D,A,1,3)) + ...
          (lb/2)*(A(19)*mvector(dX,v,v_m1,v_p1,A,3,i,j) + mvector(dX,v,v_m1,v_p1,A,1,i,j)*A(21) - 2*(mdotc)*A(3)) -
2*kk*(A(12) - A(3)*trB) - 6*C_I * gamma_mag * (A(3) - (1/3) * I(3)));

dA(4) = dA(2);

dA(5) = OF*(((W(4)*A(2) + W(5)*A(5) + W(6)*A(8)) - (A(4)*W(2) + A(5)*W(5) + A(6)*W(8))) + ((D(4)*A(2) + D(5)*A(5) +
D(6)*A(8)) + (A(4)*D(2) + A(5)*D(5) + A(6)*D(8)) - 2*Explicit_IBOF_v(D,A,2,2)) + ...
          (lb/2)*(A(20)*mvector(dX,v,v_m1,v_p1,A,2,i,j) + mvector(dX,v,v_m1,v_p1,A,2,i,j)*A(20) - 2*(mdotc)*A(5)) -
2*kk*(A(14) - A(5)*trB) - 6*C_I * gamma_mag * (A(5) - (1/3) * I(5)));

dA(6) = OF*(((W(4)*A(3) + W(5)*A(6) + W(6)*A(9)) - (A(4)*W(3) + A(5)*W(6) + A(6)*W(9))) + ((D(4)*A(3) + D(5)*A(6) +
D(6)*A(9)) + (A(4)*D(3) + A(5)*D(6) + A(6)*D(9)) - 2*Explicit_IBOF_v(D,A,2,3)) + ...
          (lb/2)*(A(20)*mvector(dX,v,v_m1,v_p1,A,3,i,j) + mvector(dX,v,v_m1,v_p1,A,2,i,j)*A(21) - 2*(mdotc)*A(6)) -
2*kk*(A(15) - A(6)*trB) - 6*C_I * gamma_mag * (A(6) - (1/3) * I(6)));
```

```
dA(7) = dA(3);

dA(8) = dA(6);

dA(9) = OF*(((W(7)*A(3) + W(8)*A(6) + W(9)*A(9)) - (A(7)*W(3) + A(8)*W(6) + A(9)*W(9))) + ((D(7)*A(3) + D(8)*A(6) +
D(9)*A(9)) + (A(7)*D(3) + A(8)*D(6) + A(9)*D(9)) - 2*Explicit_IBOF_v(D,A,3,3)) + ...
        (lb/2)*(A(21)*mvector(dX,v,v_m1,v_p1,A,3,i,j) + mvector(dX,v,v_m1,v_p1,A,3,i,j)*A(21) - 2*(mdotc)*A(9)) -
2*kk*(A(18) - A(9)*trB) - 6*C_I * gamma_mag * (A(9) - (1/3) * I(9)));

%% B Orientation Tensor Calculations

dA(10) = OF*(((W(1)*A(10) + W(2)*A(13) + W(3)*A(16)) - (A(10)*W(1) + A(11)*W(4) + A(12)*W(7))) + ((D(1)*A(10) +
D(2)*A(13) + D(3)*A(16)) + (A(10)*D(1) + A(11)*D(4) + A(12)*D(7)) - ddoublea*A(10)) + ...
        (lb/2)*(A(19)*mvector(dX,v,v_m1,v_p1,A,1,i,j) + mvector(dX,v,v_m1,v_p1,A,1,i,j)*A(19) - 2*(mdotc)*A(10)) -
2*kk*(A(1) - A(10)*trB) - 4*C_I * gamma_mag * A(10));

dA(11) = OF*(((W(1)*A(11) + W(2)*A(14) + W(3)*A(17)) - (A(10)*W(2) + A(11)*W(5) + A(12)*W(8))) + ((D(1)*A(11) +
D(2)*A(14) + D(3)*A(17)) + (A(10)*D(2) + A(11)*D(5) + A(12)*D(8)) - ddoublea*A(11)) + ...
        (lb/2)*(A(19)*mvector(dX,v,v_m1,v_p1,A,2,i,j) + mvector(dX,v,v_m1,v_p1,A,1,i,j)*A(20) - 2*(mdotc)*A(11)) -
2*kk*(A(2) - A(11)*trB) - 4*C_I * gamma_mag * A(11));

dA(12) = OF*(((W(1)*A(12) + W(2)*A(15) + W(3)*A(18)) - (A(10)*W(3) + A(11)*W(6) + A(12)*W(9))) + ((D(1)*A(12) +
D(2)*A(15) + D(3)*A(18)) + (A(10)*D(3) + A(11)*D(6) + A(12)*D(9)) - ddoublea*A(12)) + ...
        (lb/2)*(A(19)*mvector(dX,v,v_m1,v_p1,A,3,i,j) + mvector(dX,v,v_m1,v_p1,A,1,i,j)*A(21) - 2*(mdotc)*A(12)) -
2*kk*(A(3) - A(12)*trB) - 4*C_I * gamma_mag * A(12));

dA(13) = dA(11);

dA(14) = OF*(((W(4)*A(11) + W(5)*A(14) + W(6)*A(17)) - (A(13)*W(2) + A(14)*W(5) + A(15)*W(8))) + ((D(4)*A(11) +
D(5)*A(14) + D(6)*A(17)) + (A(13)*D(2) + A(14)*D(5) + A(15)*D(8)) - ddoublea*A(14)) + ...
        (lb/2)*(A(20)*mvector(dX,v,v_m1,v_p1,A,2,i,j) + mvector(dX,v,v_m1,v_p1,A,2,i,j)*A(20) - 2*(mdotc)*A(14)) -
2*kk*(A(5) - A(14)*trB) - 4*C_I * gamma_mag * A(14));

dA(15) = OF*(((W(4)*A(12) + W(5)*A(15) + W(6)*A(18)) - (A(13)*W(3) + A(14)*W(6) + A(15)*W(9))) + ((D(4)*A(12) +
D(5)*A(15) + D(6)*A(18)) + (A(13)*D(3) + A(14)*D(6) + A(15)*D(9)) - ddoublea*A(15)) + ...
        (lb/2)*(A(20)*mvector(dX,v,v_m1,v_p1,A,3,i,j) + mvector(dX,v,v_m1,v_p1,A,2,i,j)*A(21) - 2*(mdotc)*A(15)) -
2*kk*(A(6) - A(15)*trB) - 4*C_I * gamma_mag * A(15));

dA(16) = dA(12);

dA(17) = dA(15);

dA(18) = OF*(((W(7)*A(12) + W(8)*A(15) + W(9)*A(18)) - (A(16)*W(3) + A(17)*W(6) + A(18)*W(9))) + ((D(7)*A(12) +
D(8)*A(15) + D(9)*A(18)) + (A(16)*D(3) + A(17)*D(6) + A(18)*D(9)) - ddoublea*A(18)) + ...
        (lb/2)*(A(21)*mvector(dX,v,v_m1,v_p1,A,3,i,j) + mvector(dX,v,v_m1,v_p1,A,3,i,j)*A(21) - 2*(mdotc)*A(18)) -
2*kk*(A(9) - A(18)*trB) - 4*C_I * gamma_mag * A(18));


%% C Orientation Tensor Calculations
% Calculating A:delVt

Advidouble = A(1)*dvi(1) + A(2)*dvi(4) + A(3)*dvi(7) + ...
        A(4)*dvi(2) + A(5)*dvi(5) + A(6)*dvi(8) + ...
        A(7)*dvi(3) + A(8)*dvi(6) + A(9)*dvi(9);
```

```
dA(19) = OF*((dvi(1)*A(19) + dvi(4)*A(20) + dvi(7)*A(21)) - Advidouble*A(19) + (lb/2)*(mvector(dX,v,v_m1,v_p1,A,1,i,j) -
A(19)*(mdotc)) - kk*A(19)*(1 - trB) - 2*C_I*gamma_mag*A(19));
dA(20) = OF*((dvi(2)*A(19) + dvi(5)*A(20) + dvi(8)*A(21)) - Advidouble*A(20) + (lb/2)*(mvector(dX,v,v_m1,v_p1,A,2,i,j) -
A(20)*(mdotc)) - kk*A(20)*(1 - trB) - 2*C_I*gamma_mag*A(20));
dA(21) = OF*((dvi(3)*A(19) + dvi(6)*A(20) + dvi(9)*A(21)) - Advidouble*A(21) + (lb/2)*(mvector(dX,v,v_m1,v_p1,A,3,i,j) -
A(21)*(mdotc)) - kk*A(21)*(1 - trB) - 2*C_I*gamma_mag*A(21));
end
```

E.2.6 BR_IBOFc.m

```
function [dA] =
BR_IBOFc(dX,dvi,v,v_m1,v_p1,vv,w,d,lb,kk,C_I,OF,A,i,j,Am11,Am12,Am13,Bm11,Bm12,Bm13,Cm11,Cm12,Cm13)
%% Parameter Definitions

W = w;
D = d;
gamma = 2*D;
I = [1 0 0 0 1 0 0 0 1];

%% Double-Dot Terms
gamma_mag = sqrt((1/2)*(gamma(1)*gamma(1) + gamma(2)*gamma(2) + gamma(3)*gamma(3) + ...
                gamma(4)*gamma(4) + gamma(5)*gamma(5) + gamma(6)*gamma(6) + ...
                gamma(7)*gamma(7) + gamma(8)*gamma(8) + gamma(9)*gamma(9)));

%% Calculation of trace
trB = A(10) + A(14) + A(18);

%% Calculation of m*C
mdotc = mvector(dX,v,v_m1,v_p1,A,1,i,j)*A(19) + mvector(dX,v,v_m1,v_p1,A,2,i,j)*A(20) +
mvector(dX,v,v_m1,v_p1,A,3,i,j)*A(21);

%% Calculation of D:A
ddoublea = 2*(D(1)*A(1) + D(2)*A(4) + D(3)*A(7) + ...
          D(4)*A(2) + D(5)*A(5) + D(6)*A(8) + ...
          D(7)*A(3) + D(8)*A(6) + D(9)*A(9));

%% A Orientation Tensor Calculations

convBR = convectiveBR(A,Am11,Am12,Am13,Bm11,Bm12,Bm13,Cm11,Cm12,Cm13,vv,dX);

dA = zeros(21,1);

dA(1) = OF*(((W(1)*A(1) + W(2)*A(4) + W(3)*A(7)) - (A(1)*W(1) + A(2)*W(4) + A(3)*W(7))) + ((D(1)*A(1) + D(2)*A(4) +
D(3)*A(7)) + (A(1)*D(1) + A(2)*D(4) + A(3)*D(7)) - 2*Explicit_IBOF_v(D,A,1,1)) + ...
        (lb/2)*(A(19)*mvector(dX,v,v_m1,v_p1,A,1,i,j) + mvector(dX,v,v_m1,v_p1,A,1,i,j)*A(19) - 2*(mdotc)*A(1)) -
2*kk*(A(10) - A(1)*trB) - 6*C_I * gamma_mag * (A(1) - (1/3) * I(1))) - convBR(1);

dA(2) = OF*(((W(1)*A(2) + W(2)*A(5) + W(3)*A(8)) - (A(1)*W(2) + A(2)*W(5) + A(3)*W(8))) + ((D(1)*A(2) + D(2)*A(5) +
D(3)*A(8)) + (A(1)*D(2) + A(2)*D(5) + A(3)*D(8)) - 2*Explicit_IBOF_v(D,A,1,2)) + ...
        (lb/2)*(A(19)*mvector(dX,v,v_m1,v_p1,A,2,i,j) + mvector(dX,v,v_m1,v_p1,A,1,i,j)*A(20) - 2*(mdotc)*A(2)) -
2*kk*(A(11) - A(2)*trB) - 6*C_I * gamma_mag * (A(2) - (1/3) * I(2))) - convBR(2);

dA(3) = OF*(((W(1)*A(3) + W(2)*A(6) + W(3)*A(9)) - (A(1)*W(3) + A(2)*W(6) + A(3)*W(9))) + ((D(1)*A(3) + D(2)*A(6) +
D(3)*A(9)) + (A(1)*D(3) + A(2)*D(6) + A(3)*D(9)) - 2*Explicit_IBOF_v(D,A,1,3)) + ...
        (lb/2)*(A(19)*mvector(dX,v,v_m1,v_p1,A,3,i,j) + mvector(dX,v,v_m1,v_p1,A,1,i,j)*A(21) - 2*(mdotc)*A(3)) -
2*kk*(A(12) - A(3)*trB) - 6*C_I * gamma_mag * (A(3) - (1/3) * I(3))) - convBR(3);
```

```
dA(4) = dA(2);

dA(5) = OF*(((W(4)*A(2) + W(5)*A(5) + W(6)*A(8)) - (A(4)*W(2) + A(5)*W(5) + A(6)*W(8))) + ((D(4)*A(2) + D(5)*A(5) +
D(6)*A(8)) + (A(4)*D(2) + A(5)*D(5) + A(6)*D(8)) - 2*Explicit_IBOF_v(D,A,2,2)) + ...
        (lb/2)*(A(20)*mvector(dX,v,v_m1,v_p1,A,2,i,j) + mvector(dX,v,v_m1,v_p1,A,2,i,j)*A(20) - 2*(mdotc)*A(5)) -
2*kk*(A(14) - A(5)*trB) - 6*C_I * gamma_mag * (A(5) - (1/3) * I(5))) - convBR(5);

dA(6) = OF*(((W(4)*A(3) + W(5)*A(6) + W(6)*A(9)) - (A(4)*W(3) + A(5)*W(6) + A(6)*W(9))) + ((D(4)*A(3) + D(5)*A(6) +
D(6)*A(9)) + (A(4)*D(3) + A(5)*D(6) + A(6)*D(9)) - 2*Explicit_IBOF_v(D,A,2,3)) + ...
        (lb/2)*(A(20)*mvector(dX,v,v_m1,v_p1,A,3,i,j) + mvector(dX,v,v_m1,v_p1,A,2,i,j)*A(21) - 2*(mdotc)*A(6)) -
2*kk*(A(15) - A(6)*trB) - 6*C_I * gamma_mag * (A(6) - (1/3) * I(6))) - convBR(6);

dA(7) = dA(3);

dA(8) = dA(6);

dA(9) = OF*(((W(7)*A(3) + W(8)*A(6) + W(9)*A(9)) - (A(7)*W(3) + A(8)*W(6) + A(9)*W(9))) + ((D(7)*A(3) + D(8)*A(6) +
D(9)*A(9)) + (A(7)*D(3) + A(8)*D(6) + A(9)*D(9)) - 2*Explicit_IBOF_v(D,A,3,3)) + ...
        (lb/2)*(A(21)*mvector(dX,v,v_m1,v_p1,A,3,i,j) + mvector(dX,v,v_m1,v_p1,A,3,i,j)*A(21) - 2*(mdotc)*A(9)) -
2*kk*(A(18) - A(9)*trB) - 6*C_I * gamma_mag * (A(9) - (1/3) * I(9))) - convBR(9);

%% B Orientation Tensor Calculations

dA(10) = OF*(((W(1)*A(10) + W(2)*A(13) + W(3)*A(16)) - (A(10)*W(1) + A(11)*W(4) + A(12)*W(7))) + ((D(1)*A(10) +
D(2)*A(13) + D(3)*A(16)) + (A(10)*D(1) + A(11)*D(4) + A(12)*D(7)) - ddoublea*A(10)) + ...
        (lb/2)*(A(19)*mvector(dX,v,v_m1,v_p1,A,1,i,j) + mvector(dX,v,v_m1,v_p1,A,1,i,j)*A(19) - 2*(mdotc)*A(10)) -
2*kk*(A(1) - A(10)*trB) - 4*C_I * gamma_mag * A(10)) - convBR(10);

dA(11) = OF*(((W(1)*A(11) + W(2)*A(14) + W(3)*A(17)) - (A(10)*W(2) + A(11)*W(5) + A(12)*W(8))) + ((D(1)*A(11) +
D(2)*A(14) + D(3)*A(17)) + (A(10)*D(2) + A(11)*D(5) + A(12)*D(8)) - ddoublea*A(11)) + ...
        (lb/2)*(A(19)*mvector(dX,v,v_m1,v_p1,A,2,i,j) + mvector(dX,v,v_m1,v_p1,A,1,i,j)*A(20) - 2*(mdotc)*A(11)) -
2*kk*(A(2) - A(11)*trB) - 4*C_I * gamma_mag * A(11)) - convBR(11);

dA(12) = OF*(((W(1)*A(12) + W(2)*A(15) + W(3)*A(18)) - (A(10)*W(3) + A(11)*W(6) + A(12)*W(9))) + ((D(1)*A(12) +
D(2)*A(15) + D(3)*A(18)) + (A(10)*D(3) + A(11)*D(6) + A(12)*D(9)) - ddoublea*A(12)) + ...
        (lb/2)*(A(19)*mvector(dX,v,v_m1,v_p1,A,3,i,j) + mvector(dX,v,v_m1,v_p1,A,1,i,j)*A(21) - 2*(mdotc)*A(12)) -
2*kk*(A(3) - A(12)*trB) - 4*C_I * gamma_mag * A(12)) - convBR(12);

dA(13) = dA(11);

dA(14) = OF*(((W(4)*A(11) + W(5)*A(14) + W(6)*A(17)) - (A(13)*W(2) + A(14)*W(5) + A(15)*W(8))) + ((D(4)*A(11) +
D(5)*A(14) + D(6)*A(17)) + (A(13)*D(2) + A(14)*D(5) + A(15)*D(8)) - ddoublea*A(14)) + ...
        (lb/2)*(A(20)*mvector(dX,v,v_m1,v_p1,A,2,i,j) + mvector(dX,v,v_m1,v_p1,A,2,i,j)*A(20) - 2*(mdotc)*A(14)) -
2*kk*(A(5) - A(14)*trB) - 4*C_I * gamma_mag * A(14)) - convBR(14);

dA(15) = OF*(((W(4)*A(12) + W(5)*A(15) + W(6)*A(18)) - (A(13)*W(3) + A(14)*W(6) + A(15)*W(9))) + ((D(4)*A(12) +
D(5)*A(15) + D(6)*A(18)) + (A(13)*D(3) + A(14)*D(6) + A(15)*D(9)) - ddoublea*A(15)) + ...
        (lb/2)*(A(20)*mvector(dX,v,v_m1,v_p1,A,3,i,j) + mvector(dX,v,v_m1,v_p1,A,2,i,j)*A(21) - 2*(mdotc)*A(15)) -
2*kk*(A(6) - A(15)*trB) - 4*C_I * gamma_mag * A(15)) - convBR(15);

dA(16) = dA(12);

dA(17) = dA(15);
```

```
dA(18) = OF*(((W(7)*A(12) + W(8)*A(15) + W(9)*A(18)) - (A(16)*W(3) + A(17)*W(6) + A(18)*W(9))) + ((D(7)*A(12) +
D(8)*A(15) + D(9)*A(18)) + (A(16)*D(3) + A(17)*D(6) + A(18)*D(9)) - ddoublea*A(18)) + ...
        (lb/2)*(A(21)*mvector(dX,v,v_m1,v_p1,A,3,i,j) + mvector(dX,v,v_m1,v_p1,A,3,i,j)*A(21) - 2*(mdotc)*A(18)) -
2*kk*(A(9) - A(18)*trB) - 4*C_I * gamma_mag * A(18)) - convBR(18);


%% C Orientation Tensor Calculations
% Calculating A:delVt

Advidouble = A(1)*dvi(1) + A(2)*dvi(4) + A(3)*dvi(7) + ...
        A(4)*dvi(2) + A(5)*dvi(5) + A(6)*dvi(8) + ...
        A(7)*dvi(3) + A(8)*dvi(6) + A(9)*dvi(9);

dA(19) = OF*((dvi(1)*A(19) + dvi(4)*A(20) + dvi(7)*A(21)) - Advidouble*A(19) + (lb/2)*(mvector(dX,v,v_m1,v_p1,A,1,i,j) -
A(19)*(mdotc)) - kk*A(19)*(1 - trB) - 2*C_I*gamma_mag*A(19)) - convBR(19);
dA(20) = OF*((dvi(2)*A(19) + dvi(5)*A(20) + dvi(8)*A(21)) - Advidouble*A(20) + (lb/2)*(mvector(dX,v,v_m1,v_p1,A,2,i,j) -
A(20)*(mdotc)) - kk*A(20)*(1 - trB) - 2*C_I*gamma_mag*A(20)) - convBR(20);
dA(21) = OF*((dvi(3)*A(19) + dvi(6)*A(20) + dvi(9)*A(21)) - Advidouble*A(21) + (lb/2)*(mvector(dX,v,v_m1,v_p1,A,3,i,j) -
A(21)*(mdotc)) - kk*A(21)*(1 - trB) - 2*C_I*gamma_mag*A(21)) - convBR(21);
end
```

E.2.7 RSC_quad.m

```
function [dA] = RSC_quad(w,d,C_I,OF,A)
%% Parameter Definitions

W = w;
D = d;
gamma = 2*D;
I = [1 0 0 0 1 0 0 0 1];

%% Double-Dot Terms
gamma_mag = sqrt((1/2)*(gamma(1)*gamma(1) + gamma(2)*gamma(4) + gamma(3)*gamma(7) + ...
                gamma(4)*gamma(2) + gamma(5)*gamma(5) + gamma(6)*gamma(8) + ...
                gamma(7)*gamma(3) + gamma(8)*gamma(6) + gamma(9)*gamma(9)));
%% Orientation Calculations

LD = LddD(D,A);

MAD = MddAddD(D,A);

dA = zeros(9,1);

dA(1) = (W(1)*A(1) + W(2)*A(2) + W(3)*A(3)) - (A(1)*W(1) + A(2)*W(4) + A(3)*W(7)) + ((D(1)*A(1) + D(2)*A(2) +
D(3)*A(3)) + (A(1)*D(1) + A(2)*D(4) + A(3)*D(7)) - ...
    2*(A(1)*Explicit_Quad_v(D,A) + (1-OF)*(LD(1) - MAD(1)))) + 2 * C_I * OF * gamma_mag * (I(1) - 3 * A(1));

dA(2) = (W(1)*A(2) + W(2)*A(5) + W(3)*A(6)) - (A(1)*W(2) + A(2)*W(5) + A(3)*W(8)) + ((D(1)*A(2) + D(2)*A(5) +
D(3)*A(6)) + (A(1)*D(2) + A(2)*D(5) + A(3)*D(8)) - ...
    2*(A(2)*Explicit_Quad_v(D,A) + (1-OF)*(LD(2) - MAD(2)))) + 2 * C_I * OF * gamma_mag * (I(2) - 3 * A(2));

dA(3) = (W(1)*A(3) + W(2)*A(6) + W(3)*A(9)) - (A(1)*W(3) + A(2)*W(6) + A(3)*W(9)) + ((D(1)*A(3) + D(2)*A(6) +
D(3)*A(9)) + (A(1)*D(3) + A(2)*D(6) + A(3)*D(9)) - ...
    2*(A(3)*Explicit_Quad_v(D,A) + (1-OF)*(LD(3) - MAD(3)))) + 2 * C_I * OF * gamma_mag * (I(3) - 3 * A(3));

dA(4) = dA(2);
```

```
dA(5) = (W(4)*A(2) + W(5)*A(5) + W(6)*A(6)) - (A(2)*W(2) + A(5)*W(5) + A(6)*W(8)) + ((D(4)*A(2) + D(5)*A(5) +
D(6)*A(6)) + (A(2)*D(2) + A(5)*D(5) + A(6)*D(8)) - ...
      2*(A(5)*Explicit_Quad_v(D,A) + (1-OF)*(LD(5) - MAD(5)))) + 2 * C_I * OF * gamma_mag * (I(5) - 3 * A(5));

dA(6) = (W(4)*A(3) + W(5)*A(6) + W(6)*A(9)) - (A(2)*W(3) + A(5)*W(6) + A(6)*W(9)) + ((D(4)*A(3) + D(5)*A(6) +
D(6)*A(9)) + (A(2)*D(3) + A(5)*D(6) + A(6)*D(9)) - ...
      2*(A(6)*Explicit_Quad_v(D,A) + (1-OF)*(LD(6) - MAD(6)))) + 2 * C_I * OF * gamma_mag * (I(6) - 3 * A(6));

dA(7) = dA(3);

dA(8) = dA(6);

dA(9) = (W(7)*A(3) + W(8)*A(6) + W(9)*A(9)) - (A(3)*W(3) + A(6)*W(6) + A(9)*W(9)) + ((D(7)*A(3) + D(8)*A(6) +
D(9)*A(9)) + (A(3)*D(3) + A(6)*D(6) + A(9)*D(9)) - ...
      2*(A(9)*Explicit_Quad_v(D,A) + (1-OF)*(LD(9) - MAD(9)))) + 2 * C_I * OF * gamma_mag * (I(9) - 3 * A(9));
```

E.2.8 RSC_quadc.m

```
function [dA] = RSC_quadc(w,d,C_I,OF,A,Am11,Am12,Am13,dX,v)
%% Parameter Definitions

W = w;
D = d;
gamma = 2*D;
I = [1 0 0 0 1 0 0 0 1];

%% Double-Dot Terms
gamma_mag = sqrt((1/2)*(gamma(1)*gamma(1) + gamma(2)*gamma(4) + gamma(3)*gamma(7) + ...
                  gamma(4)*gamma(2) + gamma(5)*gamma(5) + gamma(6)*gamma(8) + ...
                  gamma(7)*gamma(3) + gamma(8)*gamma(6) + gamma(9)*gamma(9)));
%% Orientation Calculations

LD = LddD(D,A);

MAD = MddAddD(D,A);

convA = convective(A,Am11,Am12,Am13,v,dX);

dA = zeros(9,1);

dA(1) = real((W(1)*A(1) + W(2)*A(2) + W(3)*A(3)) - (A(1)*W(1) + A(2)*W(4) + A(3)*W(7)) + ((D(1)*A(1) + D(2)*A(2) +
D(3)*A(3)) + (A(1)*D(1) + A(2)*D(4) + A(3)*D(7)) - ...
      2*(A(1)*Explicit_Quad_v(D,A) + (1-OF)*(LD(1) - MAD(1)))) + 2 * C_I * OF * gamma_mag * (I(1) - 3 * A(1)) -
convA(1));

dA(2) = real((W(1)*A(2) + W(2)*A(5) + W(3)*A(6)) - (A(1)*W(2) + A(2)*W(5) + A(3)*W(8)) + ((D(1)*A(2) + D(2)*A(5) +
D(3)*A(6)) + (A(1)*D(2) + A(2)*D(5) + A(3)*D(8)) - ...
      2*(A(2)*Explicit_Quad_v(D,A) + (1-OF)*(LD(2) - MAD(2)))) + 2 * C_I * OF * gamma_mag * (I(2) - 3 * A(2)) -
convA(2));

dA(3) = real((W(1)*A(3) + W(2)*A(6) + W(3)*A(9)) - (A(1)*W(3) + A(2)*W(6) + A(3)*W(9)) + ((D(1)*A(3) + D(2)*A(6) +
D(3)*A(9)) + (A(1)*D(3) + A(2)*D(6) + A(3)*D(9)) - ...
      2*(A(3)*Explicit_Quad_v(D,A) + (1-OF)*(LD(3) - MAD(3)))) + 2 * C_I * OF * gamma_mag * (I(3) - 3 * A(3)) -
convA(3));

dA(4) = dA(2);
```

```
dA(5) = real((W(4)*A(2) + W(5)*A(5) + W(6)*A(6)) - (A(2)*W(2) + A(5)*W(5) + A(6)*W(8)) + ((D(4)*A(2) + D(5)*A(5) + D(6)*A(6)) + (A(2)*D(2) + A(5)*D(5) + A(6)*D(8)) - ...
      2*(A(5)*Explicit_Quad_v(D,A) + (1-OF)*(LD(5) - MAD(5)))) + 2 * C_I * OF * gamma_mag * (I(5) - 3 * A(5)) - convA(5));

dA(6) = real((W(4)*A(3) + W(5)*A(6) + W(6)*A(9)) - (A(2)*W(3) + A(5)*W(6) + A(6)*W(9)) + ((D(4)*A(3) + D(5)*A(6) + D(6)*A(9)) + (A(2)*D(3) + A(5)*D(6) + A(6)*D(9)) - ...
      2*(A(6)*Explicit_Quad_v(D,A) + (1-OF)*(LD(6) - MAD(6)))) + 2 * C_I * OF * gamma_mag * (I(6) - 3 * A(6)) - convA(6));

dA(7) = dA(3);

dA(8) = dA(6);

dA(9) = real((W(7)*A(3) + W(8)*A(6) + W(9)*A(9)) - (A(3)*W(3) + A(6)*W(6) + A(9)*W(9)) + ((D(7)*A(3) + D(8)*A(6) + D(9)*A(9)) + (A(3)*D(3) + A(6)*D(6) + A(9)*D(9)) - ...
      2*(A(9)*Explicit_Quad_v(D,A) + (1-OF)*(LD(9) - MAD(9)))) + 2 * C_I * OF * gamma_mag * (I(9) - 3 * A(9)) - convA(9));
end
```

E.2.9 RSC_IBOF.m

```
function [dA] = RSC_IBOF(w,d,C_I,OF,A)
%% Parameter Definitions

W = w;
D = d;
gamma = 2*D;
I = [1 0 0 0 1 0 0 0 1];

%% Double-Dot Terms
gamma_mag = sqrt((1/2)*(gamma(1)*gamma(1) + gamma(2)*gamma(4) + gamma(3)*gamma(7) + ...
                  gamma(4)*gamma(2) + gamma(5)*gamma(5) + gamma(6)*gamma(8) + ...
                  gamma(7)*gamma(3) + gamma(8)*gamma(6) + gamma(9)*gamma(9)));
%% Orientation Calculations

LD = LddD(D,A);

MAD = MddAddD_IBOF(D,A);

dA = zeros(9,1);

dA(1) = real((W(1)*A(1) + W(2)*A(2) + W(3)*A(3)) - (A(1)*W(1) + A(2)*W(4) + A(3)*W(7)) + ((D(1)*A(1) + D(2)*A(2) + D(3)*A(3)) + (A(1)*D(1) + A(2)*D(4) + A(3)*D(7)) - ...
      2*(Explicit_IBOF_v(D,A,1,1) + (1-OF)*(LD(1) - MAD(1)))) + 2 * C_I * OF * gamma_mag * (I(1) - 3 * A(1)));

dA(2) = real((W(1)*A(2) + W(2)*A(5) + W(3)*A(6)) - (A(1)*W(2) + A(2)*W(5) + A(3)*W(8)) + ((D(1)*A(2) + D(2)*A(5) + D(3)*A(6)) + (A(1)*D(2) + A(2)*D(5) + A(3)*D(8)) - ...
      2*(Explicit_IBOF_v(D,A,1,2) + (1-OF)*(LD(2) - MAD(2)))) + 2 * C_I * OF * gamma_mag * (I(2) - 3 * A(2)));

dA(3) = real((W(1)*A(3) + W(2)*A(6) + W(3)*A(9)) - (A(1)*W(3) + A(2)*W(6) + A(3)*W(9)) + ((D(1)*A(3) + D(2)*A(6) + D(3)*A(9)) + (A(1)*D(3) + A(2)*D(6) + A(3)*D(9)) - ...
      2*(Explicit_IBOF_v(D,A,1,3) + (1-OF)*(LD(3) - MAD(3)))) + 2 * C_I * OF * gamma_mag * (I(3) - 3 * A(3)));

dA(4) = dA(2);
```

```
dA(5) = real((W(4)*A(2) + W(5)*A(5) + W(6)*A(6)) - (A(2)*W(2) + A(5)*W(5) + A(6)*W(8)) + ((D(4)*A(2) + D(5)*A(5) + D(6)*A(6)) + (A(2)*D(2) + A(5)*D(5) + A(6)*D(8)) - ...
    2*(Explicit_IBOF_v(D,A,2,2) + (1-OF)*(LD(5) - MAD(5)))) + 2 * C_I * OF * gamma_mag * (I(5) - 3 * A(5)));

dA(6) = real((W(4)*A(3) + W(5)*A(6) + W(6)*A(9)) - (A(2)*W(3) + A(5)*W(6) + A(6)*W(9)) + ((D(4)*A(3) + D(5)*A(6) + D(6)*A(9)) + (A(2)*D(3) + A(5)*D(6) + A(6)*D(9)) - ...
    2*(Explicit_IBOF_v(D,A,2,3) + (1-OF)*(LD(6) - MAD(6)))) + 2 * C_I * OF * gamma_mag * (I(6) - 3 * A(6)));

dA(7) = dA(3);

dA(8) = dA(6);

dA(9) = real((W(7)*A(3) + W(8)*A(6) + W(9)*A(9)) - (A(3)*W(3) + A(6)*W(6) + A(9)*W(9)) + ((D(7)*A(3) + D(8)*A(6) + D(9)*A(9)) + (A(3)*D(3) + A(6)*D(6) + A(9)*D(9)) - ...
    2*(Explicit_IBOF_v(D,A,3,3) + (1-OF)*(LD(9) - MAD(9)))) + 2 * C_I * OF * gamma_mag * (I(9) - 3 * A(9)));
End
```

E.2.10 RSC_IBOFc.m

```
function [dA] = RSC_IBOFc(w,d,C_I,OF,A,Am11,Am12,Am13,dX,v)
%% Parameter Definitions

W = w;
D = d;
gamma = 2*D;
I = [1 0 0 0 1 0 0 0 1];

%% Double-Dot Terms
gamma_mag = sqrt((1/2)*(gamma(1)*gamma(1) + gamma(2)*gamma(4) + gamma(3)*gamma(7) + ...
                gamma(4)*gamma(2) + gamma(5)*gamma(5) + gamma(6)*gamma(8) + ...
                gamma(7)*gamma(3) + gamma(8)*gamma(6) + gamma(9)*gamma(9)));

%% Orientation Calculations

LD = LddD(D,A);

MAD = MddAddD_IBOF(D,A);

convA = convective(A,Am11,Am12,Am13,v,dX);

dA = zeros(9,1);

dA(1) = real((W(1)*A(1) + W(2)*A(2) + W(3)*A(3)) - (A(1)*W(1) + A(2)*W(4) + A(3)*W(7)) + ((D(1)*A(1) + D(2)*A(2) + D(3)*A(3)) + (A(1)*D(1) + A(2)*D(4) + A(3)*D(7)) - ...
    2*(Explicit_IBOF_v(D,A,1,1) + (1-OF)*(LD(1) - MAD(1)))) + 2 * C_I * OF * gamma_mag * (I(1) - 3 * A(1))) - convA(1);

dA(2) = real((W(1)*A(2) + W(2)*A(5) + W(3)*A(6)) - (A(1)*W(2) + A(2)*W(5) + A(3)*W(8)) + ((D(1)*A(2) + D(2)*A(5) + D(3)*A(6)) + (A(1)*D(2) + A(2)*D(5) + A(3)*D(8)) - ...
    2*(Explicit_IBOF_v(D,A,1,2) + (1-OF)*(LD(2) - MAD(2)))) + 2 * C_I * OF * gamma_mag * (I(2) - 3 * A(2))) - convA(2);

dA(3) = real((W(1)*A(3) + W(2)*A(6) + W(3)*A(9)) - (A(1)*W(3) + A(2)*W(6) + A(3)*W(9)) + ((D(1)*A(3) + D(2)*A(6) + D(3)*A(9)) + (A(1)*D(3) + A(2)*D(6) + A(3)*D(9)) - ...
    2*(Explicit_IBOF_v(D,A,1,3) + (1-OF)*(LD(3) - MAD(3)))) + 2 * C_I * OF * gamma_mag * (I(3) - 3 * A(3))) - convA(3);

dA(4) = dA(2);
```

```
dA(5) = real((W(4)*A(2) + W(5)*A(5) + W(6)*A(6)) - (A(2)*W(2) + A(5)*W(5) + A(6)*W(8)) + ((D(4)*A(2) + D(5)*A(5) +
D(6)*A(6)) + (A(2)*D(2) + A(5)*D(5) + A(6)*D(8)) - ...
      2*(Explicit_IBOF_v(D,A,2,2) + (1-OF)*(LD(5) - MAD(5)))) + 2 * C_I * OF * gamma_mag * (I(5) - 3 * A(5))) - convA(5);

dA(6) = real((W(4)*A(3) + W(5)*A(6) + W(6)*A(9)) - (A(2)*W(3) + A(5)*W(6) + A(6)*W(9)) + ((D(4)*A(3) + D(5)*A(6) +
D(6)*A(9)) + (A(2)*D(3) + A(5)*D(6) + A(6)*D(9)) - ...
      2*(Explicit_IBOF_v(D,A,2,3) + (1-OF)*(LD(6) - MAD(6)))) + 2 * C_I * OF * gamma_mag * (I(6) - 3 * A(6))) - convA(6);

dA(7) = dA(3);

dA(8) = dA(6);

dA(9) = real((W(7)*A(3) + W(8)*A(6) + W(9)*A(9)) - (A(3)*W(3) + A(6)*W(6) + A(9)*W(9)) + ((D(7)*A(3) + D(8)*A(6) +
D(9)*A(9)) + (A(3)*D(3) + A(6)*D(6) + A(9)*D(9)) - ...
      2*(Explicit_IBOF_v(D,A,3,3) + (1-OF)*(LD(9) - MAD(9)))) + 2 * C_I * OF * gamma_mag * (I(9) - 3 * A(9))) - convA(9);
end
```

E.2.11 ARD_RSC_quad.m

```
function [dA] = ARD_RSC_quad(w,d,OF,b1,b2,b3,b4,b5,A)
%% Parameter Definitions

W = w;
D = d;
gamma = 2*D;
I = [1 0 0 0 1 0 0 0 1];

%% Double-Dot Terms
gamma_mag = sqrt((1/2)*(gamma(1)*gamma(1) + gamma(2)*gamma(2) + gamma(3)*gamma(3) + ...
                  gamma(4)*gamma(4) + gamma(5)*gamma(5) + gamma(6)*gamma(6) + ...
                  gamma(7)*gamma(7) + gamma(8)*gamma(8) + gamma(9)*gamma(9)));
%% Orientation Calculations

% Calculating the components of the anisotropic tensor, C.

C1 = b1*I(1) + b2*A(1) + b3*A(1)^2 + (b4/gamma_mag)*D(1) + (b5/gamma_mag^2)*D(1)^2;
C2 = b1*I(2) + b2*A(2) + b3*A(2)^2 + (b4/gamma_mag)*D(2) + (b5/gamma_mag^2)*D(2)^2;
C3 = b1*I(3) + b2*A(3) + b3*A(3)^2 + (b4/gamma_mag)*D(3) + (b5/gamma_mag^2)*D(3)^2;
C4 = b1*I(4) + b2*A(4) + b3*A(4)^2 + (b4/gamma_mag)*D(4) + (b5/gamma_mag^2)*D(4)^2;
C5 = b1*I(5) + b2*A(5) + b3*A(5)^2 + (b4/gamma_mag)*D(5) + (b5/gamma_mag^2)*D(5)^2;
C6 = b1*I(6) + b2*A(6) + b3*A(6)^2 + (b4/gamma_mag)*D(6) + (b5/gamma_mag^2)*D(6)^2;
C7 = b1*I(7) + b2*A(7) + b3*A(7)^2 + (b4/gamma_mag)*D(7) + (b5/gamma_mag^2)*D(7)^2;
C8 = b1*I(8) + b2*A(8) + b3*A(8)^2 + (b4/gamma_mag)*D(8) + (b5/gamma_mag^2)*D(8)^2;
C9 = b1*I(9) + b2*A(9) + b3*A(9)^2 + (b4/gamma_mag)*D(9) + (b5/gamma_mag^2)*D(9)^2;

trC = C1 + C5 + C9;

C = [C1 C2 C3 C4 C5 C6 C7 C8 C9];

LD = LddD(D,A);

MAD = MddAddD(D,A);

MC = MddC(C,A);

LC = LddC(C,A);
```

```
MAC = MddAddC(C,A);

dA = zeros(9,1);

dA(1) = real((W(1)*A(1) + W(2)*A(2) + W(3)*A(3)) - (A(1)*W(1) + A(2)*W(4) + A(3)*W(7)) + ((D(1)*A(1) + D(2)*A(2) +
D(3)*A(3)) + (A(1)*D(1) + A(2)*D(4) + A(3)*D(7)) - ...
    2*(A(1)*Explicit_Quad_v(D,A) + (1-OF)*(LD(1) - MAD(1)))) + gamma_mag*(2*(C1 - (1-OF)*MC(1)) - 2 * OF * trC *
A(1) - ...
    5*((C1*A(1) + C2*A(2) + C3*A(3)) + (A(1)*C1 + A(2)*C4 + A(3)*C7)) + 10*(A(1)*Explicit_Quad_v(C,A) + (1-
OF)*(LC(1) - MAC(1)))));

dA(2) = real((W(1)*A(2) + W(2)*A(5) + W(3)*A(6)) - (A(1)*W(2) + A(2)*W(5) + A(3)*W(8)) + ((D(1)*A(2) + D(2)*A(5) +
D(3)*A(6)) + (A(1)*D(2) + A(2)*D(5) + A(3)*D(8)) - ...
    2*(A(2)*Explicit_Quad_v(D,A) + (1-OF)*(LD(2) - MAD(2)))) + gamma_mag*(2*(C2 - (1-OF)*MC(2)) - 2 * OF * trC *
A(2) - ...
    5*((C1*A(2) + C2*A(5) + C3*A(6)) + (A(1)*C2 + A(2)*C5 + A(3)*C8)) + 10*(A(2)*Explicit_Quad_v(C,A) + (1-
OF)*(LC(2) - MAC(2)))));

dA(3) = real((W(1)*A(3) + W(2)*A(6) + W(3)*A(9)) - (A(1)*W(3) + A(2)*W(6) + A(3)*W(9)) + ((D(1)*A(3) + D(2)*A(6) +
D(3)*A(9)) + (A(1)*D(3) + A(2)*D(6) + A(3)*D(9)) - ...
    2*(A(3)*Explicit_Quad_v(D,A) + (1-OF)*(LD(3) - MAD(3)))) + gamma_mag*(2*(C3 - (1-OF)*MC(3)) - 2 * OF * trC *
A(3) - ...
    5*((C1*A(3) + C2*A(6) + C3*A(9)) + (A(1)*C3 + A(2)*C6 + A(3)*C9)) + 10*(A(3)*Explicit_Quad_v(C,A) + (1-
OF)*(LC(3) - MAC(3)))));

dA(4) = dA(2);

dA(5) = real((W(4)*A(2) + W(5)*A(5) + W(6)*A(6)) - (A(2)*W(2) + A(5)*W(5) + A(6)*W(8)) + ((D(4)*A(2) + D(5)*A(5) +
D(6)*A(6)) + (A(2)*D(2) + A(5)*D(5) + A(6)*D(8)) - ...
    2*(A(5)*Explicit_Quad_v(D,A) + (1-OF)*(LD(5) - MAD(5)))) + gamma_mag*(2*(C5 - (1-OF)*MC(5)) - 2 * OF * trC *
A(5) - ...
    5*((C4*A(2) + C5*A(5) + C6*A(6)) + (A(2)*C2 + A(5)*C5 + A(6)*C8)) + 10*(A(5)*Explicit_Quad_v(C,A) + (1-
OF)*(LC(5) - MAC(5)))));

dA(6) = real((W(4)*A(3) + W(5)*A(6) + W(6)*A(9)) - (A(2)*W(3) + A(5)*W(6) + A(6)*W(9)) + ((D(4)*A(3) + D(5)*A(6) +
D(6)*A(9)) + (A(2)*D(3) + A(5)*D(6) + A(6)*D(9)) - ...
    2*(A(6)*Explicit_Quad_v(D,A) + (1-OF)*(LD(6) - MAD(6)))) + gamma_mag*(2*(C6 - (1-OF)*MC(6)) - 2 * OF * trC *
A(6) - ...
    5*((C4*A(3) + C5*A(6) + C6*A(9)) + (A(2)*C3 + A(5)*C6 + A(6)*C9)) + 10*(A(6)*Explicit_Quad_v(C,A) + (1-
OF)*(LC(6) - MAC(6)))));

dA(7) = dA(3);

dA(8) = dA(6);

dA(9) = real((W(7)*A(3) + W(8)*A(6) + W(9)*A(9)) - (A(3)*W(3) + A(6)*W(6) + A(9)*W(9)) + ((D(7)*A(3) + D(8)*A(6) +
D(9)*A(9)) + (A(3)*D(3) + A(6)*D(6) + A(9)*D(9)) - ...
    2*(A(9)*Explicit_Quad_v(D,A) + (1-OF)*(LD(9) - MAD(9)))) + gamma_mag*(2*(C9 - (1-OF)*MC(9)) - 2 * OF * trC *
A(9) - ...
    5*((C7*A(3) + C8*A(6) + C9*A(9)) + (A(3)*C3 + A(6)*C6 + A(9)*C9)) + 10*(A(9)*Explicit_Quad_v(C,A) + (1-
OF)*(LC(9) - MAC(9)))));
end
```

E.2.12 ARD_RSC_quadc.m

```
function [dA] = ARD_RSC_quadc(w,d,OF,b1,b2,b3,b4,b5,A,Am11,Am12,Am13,dX,v)
```

```
%% Parameter Definitions

W = w;
D = d;
gamma = 2*D;
I = [1 0 0 0 1 0 0 0 1];

%% Double-Dot Terms
gamma_mag = sqrt((1/2)*(gamma(1)*gamma(1) + gamma(2)*gamma(2) + gamma(3)*gamma(3) + ...
                    gamma(4)*gamma(4) + gamma(5)*gamma(5) + gamma(6)*gamma(6) + ...
                    gamma(7)*gamma(7) + gamma(8)*gamma(8) + gamma(9)*gamma(9)));

%% Orientation Calculations

% Calculating the components of the anisotropic tensor, C.

C1 = b1*I(1) + b2*A(1) + b3*A(1)^2 + (b4/gamma_mag)*D(1) + (b5/gamma_mag^2)*D(1)^2;
C2 = b1*I(2) + b2*A(2) + b3*A(2)^2 + (b4/gamma_mag)*D(2) + (b5/gamma_mag^2)*D(2)^2;
C3 = b1*I(3) + b2*A(3) + b3*A(3)^2 + (b4/gamma_mag)*D(3) + (b5/gamma_mag^2)*D(3)^2;
C4 = b1*I(4) + b2*A(4) + b3*A(4)^2 + (b4/gamma_mag)*D(4) + (b5/gamma_mag^2)*D(4)^2;
C5 = b1*I(5) + b2*A(5) + b3*A(5)^2 + (b4/gamma_mag)*D(5) + (b5/gamma_mag^2)*D(5)^2;
C6 = b1*I(6) + b2*A(6) + b3*A(6)^2 + (b4/gamma_mag)*D(6) + (b5/gamma_mag^2)*D(6)^2;
C7 = b1*I(7) + b2*A(7) + b3*A(7)^2 + (b4/gamma_mag)*D(7) + (b5/gamma_mag^2)*D(7)^2;
C8 = b1*I(8) + b2*A(8) + b3*A(8)^2 + (b4/gamma_mag)*D(8) + (b5/gamma_mag^2)*D(8)^2;
C9 = b1*I(9) + b2*A(9) + b3*A(9)^2 + (b4/gamma_mag)*D(9) + (b5/gamma_mag^2)*D(9)^2;

trC = C1 + C5 + C9;

C = [C1 C2 C3 C4 C5 C6 C7 C8 C9];

LD = LddD(D,A);

MAD = MddAddD(D,A);

MC = MddC(C,A);

LC = LddC(C,A);

MAC = MddAddC(C,A);

convA = convective(A,Am11,Am12,Am13,v,dX);

dA = zeros(9,1);

dA(1) = real((W(1)*A(1) + W(2)*A(2) + W(3)*A(3)) - (A(1)*W(1) + A(2)*W(4) + A(3)*W(7)) + ((D(1)*A(1) + D(2)*A(2) +
D(3)*A(3)) + (A(1)*D(1) + A(2)*D(4) + A(3)*D(7)) - ...
    2*(A(1)*Explicit_Quad_v(D,A) + (1-OF)*(LD(1) - MAD(1)))) + gamma_mag*(2*(C1 - (1-OF)*MC(1)) - 2 * OF * trC *
A(1) - ...
    5*((C1*A(1) + C2*A(2) + C3*A(3)) + (A(1)*C1 + A(2)*C4 + A(3)*C7)) + 10*(A(1)*Explicit_Quad_v(C,A) + (1-
OF)*(LC(1) - MAC(1)))) - convA(1);

dA(2) = real((W(1)*A(2) + W(2)*A(5) + W(3)*A(6)) - (A(1)*W(2) + A(2)*W(5) + A(3)*W(8)) + ((D(1)*A(2) + D(2)*A(5) +
D(3)*A(6)) + (A(1)*D(2) + A(2)*D(5) + A(3)*D(8)) - ...
    2*(A(2)*Explicit_Quad_v(D,A) + (1-OF)*(LD(2) - MAD(2)))) + gamma_mag*(2*(C2 - (1-OF)*MC(2)) - 2 * OF * trC *
A(2) - ...
```

```
    5*((C1*A(2) + C2*A(5) + C3*A(6)) + (A(1)*C2 + A(2)*C5 + A(3)*C8)) + 10*(A(2)*Explicit_Quad_v(C,A) + (1-
OF)*(LC(2) - MAC(2)))) - convA(2);

dA(3) = real((W(1)*A(3) + W(2)*A(6) + W(3)*A(9)) - (A(1)*W(3) + A(2)*W(6) + A(3)*W(9)) + ((D(1)*A(3) + D(2)*A(6) +
D(3)*A(9)) + (A(1)*D(3) + A(2)*D(6) + A(3)*D(9)) - ...
    2*(A(3)*Explicit_Quad_v(D,A) + (1-OF)*(LD(3) - MAD(3)))) + gamma_mag*(2*(C3 - (1-OF)*MC(3)) - 2 * OF * trC *
A(3) - ...
    5*((C1*A(3) + C2*A(6) + C3*A(9)) + (A(1)*C3 + A(2)*C6 + A(3)*C9)) + 10*(A(3)*Explicit_Quad_v(C,A) + (1-
OF)*(LC(3) - MAC(3)))) - convA(3);

dA(4) = dA(2);

dA(5) = real((W(4)*A(2) + W(5)*A(5) + W(6)*A(6)) - (A(2)*W(2) + A(5)*W(5) + A(6)*W(8)) + ((D(4)*A(2) + D(5)*A(5) +
D(6)*A(6)) + (A(2)*D(2) + A(5)*D(5) + A(6)*D(8)) - ...
    2*(A(5)*Explicit_Quad_v(D,A) + (1-OF)*(LD(5) - MAD(5)))) + gamma_mag*(2*(C5 - (1-OF)*MC(5)) - 2 * OF * trC *
A(5) - ...
    5*((C4*A(2) + C5*A(5) + C6*A(6)) + (A(2)*C2 + A(5)*C5 + A(6)*C8)) + 10*(A(5)*Explicit_Quad_v(C,A) + (1-
OF)*(LC(5) - MAC(5)))) - convA(5);

dA(6) = real((W(4)*A(3) + W(5)*A(6) + W(6)*A(9)) - (A(2)*W(3) + A(5)*W(6) + A(6)*W(9)) + ((D(4)*A(3) + D(5)*A(6) +
D(6)*A(9)) + (A(2)*D(3) + A(5)*D(6) + A(6)*D(9)) - ...
    2*(A(6)*Explicit_Quad_v(D,A) + (1-OF)*(LD(6) - MAD(6)))) + gamma_mag*(2*(C6 - (1-OF)*MC(6)) - 2 * OF * trC *
A(6) - ...
    5*((C4*A(3) + C5*A(6) + C6*A(9)) + (A(2)*C3 + A(5)*C6 + A(6)*C9)) + 10*(A(6)*Explicit_Quad_v(C,A) + (1-
OF)*(LC(6) - MAC(6)))) - convA(6);

dA(7) = dA(3);

dA(8) = dA(6);

dA(9) = real((W(7)*A(3) + W(8)*A(6) + W(9)*A(9)) - (A(3)*W(3) + A(6)*W(6) + A(9)*W(9)) + ((D(7)*A(3) + D(8)*A(6) +
D(9)*A(9)) + (A(3)*D(3) + A(6)*D(6) + A(9)*D(9)) - ...
    2*(A(9)*Explicit_Quad_v(D,A) + (1-OF)*(LD(9) - MAD(9)))) + gamma_mag*(2*(C9 - (1-OF)*MC(9)) - 2 * OF * trC *
A(9) - ...
    5*((C7*A(3) + C8*A(6) + C9*A(9)) + (A(3)*C3 + A(6)*C6 + A(9)*C9)) + 10*(A(9)*Explicit_Quad_v(C,A) + (1-
OF)*(LC(9) - MAC(9)))) - convA(9);
End
```

E.2.13 ARD_RSC_IBOF.m

```
function [dA] = ARD_RSC_IBOF(w,d,OF,b1,b2,b3,b4,b5,A)
%% Parameter Definitions

W = w;
D = d;
gamma = 2*D;
I = [1 0 0 0 1 0 0 0 1];

%% Double-Dot Terms
gamma_mag = sqrt((1/2)*(gamma(1)*gamma(1) + gamma(2)*gamma(2) + gamma(3)*gamma(3) + ...
                gamma(4)*gamma(4) + gamma(5)*gamma(5) + gamma(6)*gamma(6) + ...
                gamma(7)*gamma(7) + gamma(8)*gamma(8) + gamma(9)*gamma(9)));
%% Orientation Calculations

% Calculating the components of the anisotropic tensor, C.

C(1) = b1*I(1) + b2*A(1) + b3*A(1)^2 + (b4/gamma_mag)*D(1) + (b5/gamma_mag^2)*D(1)^2;
C(2) = b1*I(2) + b2*A(2) + b3*A(2)^2 + (b4/gamma_mag)*D(2) + (b5/gamma_mag^2)*D(2)^2;
```

```
C(3) = b1*I(3) + b2*A(3) + b3*A(3)^2 + (b4/gamma_mag)*D(3) + (b5/gamma_mag^2)*D(3)^2;
C(4) = b1*I(4) + b2*A(4) + b3*A(4)^2 + (b4/gamma_mag)*D(4) + (b5/gamma_mag^2)*D(4)^2;
C(5) = b1*I(5) + b2*A(5) + b3*A(5)^2 + (b4/gamma_mag)*D(5) + (b5/gamma_mag^2)*D(5)^2;
C(6) = b1*I(6) + b2*A(6) + b3*A(6)^2 + (b4/gamma_mag)*D(6) + (b5/gamma_mag^2)*D(6)^2;
C(7) = b1*I(7) + b2*A(7) + b3*A(7)^2 + (b4/gamma_mag)*D(7) + (b5/gamma_mag^2)*D(7)^2;
C(8) = b1*I(8) + b2*A(8) + b3*A(8)^2 + (b4/gamma_mag)*D(8) + (b5/gamma_mag^2)*D(8)^2;
C(9) = b1*I(9) + b2*A(9) + b3*A(9)^2 + (b4/gamma_mag)*D(9) + (b5/gamma_mag^2)*D(9)^2;

trC = C(1) + C(5) + C(9);


LD = LddD(D,A);

MAD = MddAddD_IBOF(D,A);

MC = MddC(C,A);

LC = LddC(C,A);

MAC = MddAddC_IBOF(C,A);

dA = zeros(9,1);

dA(1) = real((W(1)*A(1) + W(2)*A(2) + W(3)*A(3)) - (A(1)*W(1) + A(2)*W(4) + A(3)*W(7)) + ((D(1)*A(1) + D(2)*A(2) +
D(3)*A(3)) + (A(1)*D(1) + A(2)*D(4) + A(3)*D(7)) - ...
    2*(Explicit_IBOF_v(D,A,1,1) + (1-OF)*(LD(1) - MAD(1)))) + gamma_mag*(2*(C(1) - (1-OF)*MC(1)) - 2 * OF * trC *
A(1) - ...
    5*((C(1)*A(1) + C(2)*A(2) + C(3)*A(3)) + (A(1)*C(1) + A(2)*C(4) + A(3)*C(7))) + 10*(Explicit_IBOF_v(C,A,1,1) + (1-
OF)*(LC(1) - MAC(1)))));

dA(2) = real((W(1)*A(2) + W(2)*A(5) + W(3)*A(6)) - (A(1)*W(2) + A(2)*W(5) + A(3)*W(8)) + ((D(1)*A(2) + D(2)*A(5) +
D(3)*A(6)) + (A(1)*D(2) + A(2)*D(5) + A(3)*D(8)) - ...
    2*(Explicit_IBOF_v(D,A,1,2) + (1-OF)*(LD(2) - MAD(2)))) + gamma_mag*(2*(C(2) - (1-OF)*MC(2)) - 2 * OF * trC *
A(2) - ...
    5*((C(1)*A(2) + C(2)*A(5) + C(3)*A(6)) + (A(1)*C(2) + A(2)*C(5) + A(3)*C(8))) + 10*(Explicit_IBOF_v(C,A,1,2) + (1-
OF)*(LC(2) - MAC(2)))));

dA(3) = real((W(1)*A(3) + W(2)*A(6) + W(3)*A(9)) - (A(1)*W(3) + A(2)*W(6) + A(3)*W(9)) + ((D(1)*A(3) + D(2)*A(6) +
D(3)*A(9)) + (A(1)*D(3) + A(2)*D(6) + A(3)*D(9)) - ...
    2*(Explicit_IBOF_v(D,A,1,3) + (1-OF)*(LD(3) - MAD(3)))) + gamma_mag*(2*(C(3) - (1-OF)*MC(3)) - 2 * OF * trC *
A(3) - ...
    5*((C(1)*A(3) + C(2)*A(6) + C(3)*A(9)) + (A(1)*C(3) + A(2)*C(6) + A(3)*C(9))) + 10*(Explicit_IBOF_v(C,A,1,3) + (1-
OF)*(LC(3) - MAC(3)))));

dA(4) = dA(2);

dA(5) = real((W(4)*A(2) + W(5)*A(5) + W(6)*A(6)) - (A(2)*W(2) + A(5)*W(5) + A(6)*W(8)) + ((D(4)*A(2) + D(5)*A(5) +
D(6)*A(6)) + (A(2)*D(2) + A(5)*D(5) + A(6)*D(8)) - ...
    2*(Explicit_IBOF_v(D,A,2,2) + (1-OF)*(LD(5) - MAD(5)))) + gamma_mag*(2*(C(5) - (1-OF)*MC(5)) - 2 * OF * trC *
A(5) - ...
    5*((C(4)*A(2) + C(5)*A(5) + C(6)*A(6)) + (A(2)*C(2) + A(5)*C(5) + A(6)*C(8))) + 10*(Explicit_IBOF_v(C,A,2,2) + (1-
OF)*(LC(5) - MAC(5)))));

dA(6) = real((W(4)*A(3) + W(5)*A(6) + W(6)*A(9)) - (A(2)*W(3) + A(5)*W(6) + A(6)*W(9)) + ((D(4)*A(3) + D(5)*A(6) +
D(6)*A(9)) + (A(2)*D(3) + A(5)*D(6) + A(6)*D(9)) - ...
    2*(Explicit_IBOF_v(D,A,2,3) + (1-OF)*(LD(6) - MAD(6)))) + gamma_mag*(2*(C(6) - (1-OF)*MC(6)) - 2 * OF * trC *
A(6) - ...
```

```
    5*((C(4)*A(3) + C(5)*A(6) + C(6)*A(9)) + (A(2)*C(3) + A(5)*C(6) + A(6)*C(9))) + 10*(Explicit_IBOF_v(C,A,2,3) + (1-
OF)*(LC(6) - MAC(6)))));

dA(7) = dA(3);

dA(8) = dA(6);

dA(9) = real((W(7)*A(3) + W(8)*A(6) + W(9)*A(9)) - (A(3)*W(3) + A(6)*W(6) + A(9)*W(9)) + ((D(7)*A(3) + D(8)*A(6) +
D(9)*A(9)) + (A(3)*D(3) + A(6)*D(6) + A(9)*D(9)) - ...
    2*(Explicit_IBOF_v(D,A,3,3) + (1-OF)*(LD(9) - MAD(9)))) + gamma_mag*(2*(C(9) - (1-OF)*MC(9)) - 2 * OF * trC *
A(9) - ...
    5*((C(7)*A(3) + C(8)*A(6) + C(9)*A(9)) + (A(3)*C(3) + A(6)*C(6) + A(9)*C(9))) + 10*(Explicit_IBOF_v(C,A,3,3) + (1-
OF)*(LC(9) - MAC(9)))));

end
```

E.2.14 ARD_RSC_IBOFc.m

```
function [dA] = ARD_RSC_IBOFc(w,d,OF,b1,b2,b3,b4,b5,A,Am11,Am12,Am13,dX,v)
%% Parameter Definitions

W = w;
D = d;
gamma = 2*D;
I = [1 0 0 0 1 0 0 0 1];

%% Double-Dot Terms
gamma_mag = sqrt((1/2)*(gamma(1)*gamma(1) + gamma(2)*gamma(2) + gamma(3)*gamma(3) + ...
                gamma(4)*gamma(4) + gamma(5)*gamma(5) + gamma(6)*gamma(6) + ...
                gamma(7)*gamma(7) + gamma(8)*gamma(8) + gamma(9)*gamma(9)));
%% Orientation Calculations

% Calculating the components of the anisotropic tensor, C.

C1 = b1*I(1) + b2*A(1) + b3*A(1)^2 + (b4/gamma_mag)*D(1) + (b5/gamma_mag^2)*D(1)^2;
C2 = b1*I(2) + b2*A(2) + b3*A(2)^2 + (b4/gamma_mag)*D(2) + (b5/gamma_mag^2)*D(2)^2;
C3 = b1*I(3) + b2*A(3) + b3*A(3)^2 + (b4/gamma_mag)*D(3) + (b5/gamma_mag^2)*D(3)^2;
C4 = b1*I(4) + b2*A(4) + b3*A(4)^2 + (b4/gamma_mag)*D(4) + (b5/gamma_mag^2)*D(4)^2;
C5 = b1*I(5) + b2*A(5) + b3*A(5)^2 + (b4/gamma_mag)*D(5) + (b5/gamma_mag^2)*D(5)^2;
C6 = b1*I(6) + b2*A(6) + b3*A(6)^2 + (b4/gamma_mag)*D(6) + (b5/gamma_mag^2)*D(6)^2;
C7 = b1*I(7) + b2*A(7) + b3*A(7)^2 + (b4/gamma_mag)*D(7) + (b5/gamma_mag^2)*D(7)^2;
C8 = b1*I(8) + b2*A(8) + b3*A(8)^2 + (b4/gamma_mag)*D(8) + (b5/gamma_mag^2)*D(8)^2;
C9 = b1*I(9) + b2*A(9) + b3*A(9)^2 + (b4/gamma_mag)*D(9) + (b5/gamma_mag^2)*D(9)^2;

trC = C1 + C5 + C9;

C = [C1 C2 C3 C4 C5 C6 C7 C8 C9];

LD = LddD(D,A);

MAD = MddAddD_IBOF(D,A);

MC = MddC(C,A);

LC = LddC(C,A);
```

```
MAC = MddAddC_IBOF(C,A);

convA = convective(A,Am11,Am12,Am13,v,dX);

dA = zeros(9,1);

dA(1) = real((W(1)*A(1) + W(2)*A(2) + W(3)*A(3)) - (A(1)*W(1) + A(2)*W(4) + A(3)*W(7)) + ((D(1)*A(1) + D(2)*A(2) +
D(3)*A(3)) + (A(1)*D(1) + A(2)*D(4) + A(3)*D(7)) - ...
    2*(Explicit_IBOF_v(D,A,1,1) + (1-OF)*(LD(1) - MAD(1)))) + gamma_mag*(2*(C1 - (1-OF)*MC(1)) - 2 * OF * trC *
A(1) - ...
    5*((C1*A(1) + C2*A(2) + C3*A(3)) + (A(1)*C1 + A(2)*C4 + A(3)*C7)) + 10*(Explicit_IBOF_v(C,A,1,1) + (1-
OF)*(LC(1) - MAC(1))))) - convA(1);

dA(2) = real((W(1)*A(2) + W(2)*A(5) + W(3)*A(6)) - (A(1)*W(2) + A(2)*W(5) + A(3)*W(8)) + ((D(1)*A(2) + D(2)*A(5) +
D(3)*A(6)) + (A(1)*D(2) + A(2)*D(5) + A(3)*D(8)) - ...
    2*(Explicit_IBOF_v(D,A,1,2) + (1-OF)*(LD(2) - MAD(2)))) + gamma_mag*(2*(C2 - (1-OF)*MC(2)) - 2 * OF * trC *
A(2) - ...
    5*((C1*A(2) + C2*A(5) + C3*A(6)) + (A(1)*C2 + A(2)*C5 + A(3)*C8)) + 10*(Explicit_IBOF_v(C,A,1,2) + (1-
OF)*(LC(2) - MAC(2))))) - convA(2);

dA(3) = real((W(1)*A(3) + W(2)*A(6) + W(3)*A(9)) - (A(1)*W(3) + A(2)*W(6) + A(3)*W(9)) + ((D(1)*A(3) + D(2)*A(6) +
D(3)*A(9)) + (A(1)*D(3) + A(2)*D(6) + A(3)*D(9)) - ...
    2*(Explicit_IBOF_v(D,A,1,3) + (1-OF)*(LD(3) - MAD(3)))) + gamma_mag*(2*(C3 - (1-OF)*MC(3)) - 2 * OF * trC *
A(3) - ...
    5*((C1*A(3) + C2*A(6) + C3*A(9)) + (A(1)*C3 + A(2)*C6 + A(3)*C9)) + 10*(Explicit_IBOF_v(C,A,1,3) + (1-
OF)*(LC(3) - MAC(3))))) - convA(3);

dA(4) = dA(2);

dA(5) = real((W(4)*A(2) + W(5)*A(5) + W(6)*A(6)) - (A(2)*W(2) + A(5)*W(5) + A(6)*W(8)) + ((D(4)*A(2) + D(5)*A(5) +
D(6)*A(6)) + (A(2)*D(2) + A(5)*D(5) + A(6)*D(8)) - ...
    2*(Explicit_IBOF_v(D,A,2,2) + (1-OF)*(LD(5) - MAD(5)))) + gamma_mag*(2*(C5 - (1-OF)*MC(5)) - 2 * OF * trC *
A(5) - ...
    5*((C4*A(2) + C5*A(5) + C6*A(6)) + (A(2)*C2 + A(5)*C5 + A(6)*C8)) + 10*(Explicit_IBOF_v(C,A,2,2) + (1-
OF)*(LC(5) - MAC(5))))) - convA(5);

dA(6) = real((W(4)*A(3) + W(5)*A(6) + W(6)*A(9)) - (A(2)*W(3) + A(5)*W(6) + A(6)*W(9)) + ((D(4)*A(3) + D(5)*A(6) +
D(6)*A(9)) + (A(2)*D(3) + A(5)*D(6) + A(6)*D(9)) - ...
    2*(Explicit_IBOF_v(D,A,2,3) + (1-OF)*(LD(6) - MAD(6)))) + gamma_mag*(2*(C6 - (1-OF)*MC(6)) - 2 * OF * trC *
A(6) - ...
    5*((C4*A(3) + C5*A(6) + C6*A(9)) + (A(2)*C3 + A(5)*C6 + A(6)*C9)) + 10*(Explicit_IBOF_v(C,A,2,3) + (1-
OF)*(LC(6) - MAC(6))))) - convA(6);

dA(7) = dA(3);

dA(8) = dA(6);

dA(9) = real((W(7)*A(3) + W(8)*A(6) + W(9)*A(9)) - (A(3)*W(3) + A(6)*W(6) + A(9)*W(9)) + ((D(7)*A(3) + D(8)*A(6) +
D(9)*A(9)) + (A(3)*D(3) + A(6)*D(6) + A(9)*D(9)) - ...
    2*(Explicit_IBOF_v(D,A,3,3) + (1-OF)*(LD(9) - MAD(9)))) + gamma_mag*(2*(C9 - (1-OF)*MC(9)) - 2 * OF * trC *
A(9) - ...
    5*((C7*A(3) + C8*A(6) + C9*A(9)) + (A(3)*C3 + A(6)*C6 + A(9)*C9)) + 10*(Explicit_IBOF_v(C,A,3,3) + (1-
OF)*(LC(9) - MAC(9))))) - convA(9);

end
```

E.3 Ancillary Routines

E.3.1 convective.m

```
function [convA] = convective(A,Am11,Am12,Am13,v,dX)

dA11dx1 = v(1)*((A(1) - Am11(1))/dX(1));
dA12dx1 = v(1)*((A(2) - Am11(2))/dX(1));
dA13dx1 = v(1)*((A(3) - Am11(3))/dX(1));
dA22dx1 = v(1)*((A(5) - Am11(5))/dX(1));
dA23dx1 = v(1)*((A(6) - Am11(6))/dX(1));
dA33dx1 = v(1)*((A(9) - Am11(9))/dX(1));

dA11dx2 = v(2)*((A(1) - Am12(1))/dX(2));
dA12dx2 = v(2)*((A(2) - Am12(2))/dX(2));
dA13dx2 = v(2)*((A(3) - Am12(3))/dX(2));
dA22dx2 = v(2)*((A(5) - Am12(5))/dX(2));
dA23dx2 = v(2)*((A(6) - Am12(6))/dX(2));
dA33dx2 = v(2)*((A(9) - Am12(9))/dX(2));

dA11dx3 = v(3)*((A(1) - Am13(1))/dX(3));
dA12dx3 = v(3)*((A(2) - Am13(2))/dX(3));
dA13dx3 = v(3)*((A(3) - Am13(3))/dX(3));
dA22dx3 = v(3)*((A(5) - Am13(5))/dX(3));
dA23dx3 = v(3)*((A(6) - Am13(6))/dX(3));
dA33dx3 = v(3)*((A(9) - Am13(9))/dX(3));

convA11 = dA11dx1 + dA11dx2 + dA11dx3;
convA12 = dA12dx1 + dA12dx2 + dA12dx3;
convA13 = dA13dx1 + dA13dx2 + dA13dx3;
convA22 = dA22dx1 + dA22dx2 + dA22dx3;
convA23 = dA23dx1 + dA23dx2 + dA23dx3;
convA33 = dA33dx1 + dA33dx2 + dA33dx3;

convA = [convA11 convA12 convA13 convA12 convA22 convA23 convA13 convA23 convA33];

convA(isnan(convA)) = 0;
convA(isinf(convA)) = 0;

end
```

E.3.2 convectiveBR.m

```
function [convBR] = convectiveBR(A,Am11,Am12,Am13,v,dX)
%% Convective Calculation of A

dA11dx1 = v(1)*((A(1) - Am11(1))/dX(1));
dA12dx1 = v(1)*((A(2) - Am11(2))/dX(1));
dA13dx1 = v(1)*((A(3) - Am11(3))/dX(1));
dA22dx1 = v(1)*((A(5) - Am11(5))/dX(1));
dA23dx1 = v(1)*((A(6) - Am11(6))/dX(1));
dA33dx1 = v(1)*((A(9) - Am11(9))/dX(1));

dA11dx2 = v(2)*((A(1) - Am12(1))/dX(2));
dA12dx2 = v(2)*((A(2) - Am12(2))/dX(2));
dA13dx2 = v(2)*((A(3) - Am12(3))/dX(2));
dA22dx2 = v(2)*((A(5) - Am12(5))/dX(2));
dA23dx2 = v(2)*((A(6) - Am12(6))/dX(2));
```

```
dA33dx2 = v(2)*((A(9) - Am12(9))/dX(2));

dA11dx3 = v(3)*((A(1) - Am13(1))/dX(3));
dA12dx3 = v(3)*((A(2) - Am13(2))/dX(3));
dA13dx3 = v(3)*((A(3) - Am13(3))/dX(3));
dA22dx3 = v(3)*((A(5) - Am13(5))/dX(3));
dA23dx3 = v(3)*((A(6) - Am13(6))/dX(3));
dA33dx3 = v(3)*((A(9) - Am13(9))/dX(3));

%% Convective Calculation of B

dB11dx1 = v(1)*((A(10) - Bm11(1))/dX(1));
dB12dx1 = v(1)*((A(11) - Bm11(2))/dX(1));
dB13dx1 = v(1)*((A(12) - Bm11(3))/dX(1));
dB22dx1 = v(1)*((A(14) - Bm11(5))/dX(1));
dB23dx1 = v(1)*((A(15) - Bm11(6))/dX(1));
dB33dx1 = v(1)*((A(18) - Bm11(9))/dX(1));

dB11dx2 = v(2)*((A(10) - Bm12(1))/dX(2));
dB12dx2 = v(2)*((A(11) - Bm12(2))/dX(2));
dB13dx2 = v(2)*((A(12) - Bm12(3))/dX(2));
dB22dx2 = v(2)*((A(14) - Bm12(5))/dX(2));
dB23dx2 = v(2)*((A(15) - Bm12(6))/dX(2));
dB33dx2 = v(2)*((A(18) - Bm12(9))/dX(2));

dB11dx3 = v(3)*((A(10) - Bm13(1))/dX(3));
dB12dx3 = v(3)*((A(11) - Bm13(2))/dX(3));
dB13dx3 = v(3)*((A(12) - Bm13(3))/dX(3));
dB22dx3 = v(3)*((A(14) - Bm13(5))/dX(3));
dB23dx3 = v(3)*((A(15) - Bm13(6))/dX(3));
dB33dx3 = v(3)*((A(18) - Bm13(9))/dX(3));

%% Convective Calculation of C

dC1dx1 = v(1)*((A(19) - Cm12(5))/dX(1));
dC2dx1 = v(1)*((A(20) - Cm12(6))/dX(1));
dC3dx1 = v(1)*((A(21) - Cm12(9))/dX(1));

dC1dx2 = v(2)*((A(19) - Cm13(1))/dX(2));
dC2dx2 = v(2)*((A(20) - Cm13(2))/dX(2));
dC3dx2 = v(2)*((A(21) - Cm13(3))/dX(2));

dC1dx3 = v(3)*((A(19) - Cm13(5))/dX(3));
dC2dx3 = v(3)*((A(20) - Cm13(6))/dX(3));
dC3dx3 = v(3)*((A(21) - Cm13(9))/dX(3));

%% Summation of Individual Terms to Form Tensor

convA11 = dA11dx1 + dA11dx2 + dA11dx3;
convA12 = dA12dx1 + dA12dx2 + dA12dx3;
convA13 = dA13dx1 + dA13dx2 + dA13dx3;
convA22 = dA22dx1 + dA22dx2 + dA22dx3;
convA23 = dA23dx1 + dA23dx2 + dA23dx3;
convA33 = dA33dx1 + dA33dx2 + dA33dx3;

convB11 = dB11dx1 + dB11dx2 + dB11dx3;
convB12 = dB12dx1 + dB12dx2 + dB12dx3;
convB13 = dB13dx1 + dB13dx2 + dB13dx3;
```

```
convB22 = dB22dx1 + dB22dx2 + dB22dx3;
convB23 = dB23dx1 + dB23dx2 + dB23dx3;
convB33 = dB33dx1 + dB33dx2 + dB33dx3;

convC1 = dC1dx1 + dC1dx2 + dC1dx3;
convC2 = dC2dx1 + dC2dx2 + dC2dx3;
convC3 = dC3dx1 + dC3dx2 + dC3dx3;

convA = [convA11 convA12 convA13 convA12 convA22 convA23 convA13 convA23 convA33];

convB = [convB11 convB12 convB13 convB12 convB22 convB23 convB13 convB23 convB33];

convC = [convC1 convC2 convC3];

convA(isnan(convA)) = 0;
convA(isinf(convA)) = 0;
convB(isnan(convB)) = 0;
convB(isinf(convB)) = 0;
convC(isnan(convC)) = 0;
convC(isinf(convC)) = 0;

convBR = [convA convB convC];

end
```

E.3.3 mvector.m

```
function [m] = mvector(dXbr,v,v_m1,v_p1,A0,k,i,j)

dx = dXbr(1);
dy = dXbr(2);
dz = dXbr(3);

d2V_dxidxj = zeros(9,3);

% Zeroing the entire dV2_dxidxj matrix so that it doesn't fill with NaN's
% when dx, dy, or dz are zero. (NaN is produced when dividing by zero.)

dV1_dx1dx1 = 0;
dV2_dx1dx1 = 0;
dV2_dx2dx1 = 0;
dV2_dx3dx1 = 0;
dV3_dx1dx1 = 0;
dV3_dx2dx1 = 0;
dV3_dx3dx1 = 0;
dV1_dx2dx1 = 0;
dV1_dx3dx1 = 0;

dV1_dx1dx2 = 0;
dV2_dx1dx2 = 0;
dV1_dx2dx2 = 0;
dV1_dx3dx2 = 0;
dV3_dx1dx2 = 0;
dV3_dx2dx2 = 0;
dV3_dx3dx2 = 0;
dV2_dx2dx2 = 0;
dV2_dx3dx2 = 0;
```

```
dV1_dx1dx3 = 0;
dV1_dx2dx3 = 0;
dV1_dx3dx3 = 0;
dV3_dx1dx3 = 0;
dV2_dx1dx3 = 0;
dV2_dx2dx3 = 0;
dV2_dx3dx3 = 0;
dV3_dx2dx3 = 0;
dV3_dx3dx3 = 0;

% Estimating the second derivatives using the central difference
% approximation.

if (j == 1)
%     Here we have set the j-1 term equal to the no slip condition of v = 0
   if (v_p1(j,1) ~= 0) && (v_p1(j+1,1) ~= 0)
      dV1_dx1dx1 = A110*((v_m1(j,1) - 2*v(j,1) + v_p1(j,1))/(dx^2));
      dV1_dx1dx2 = A120*((v_p1(j+1,1) - v_m1(j+1,1) - v_p1(j,1) + v_m1(j,1))/(4*dx*dy));
      dV1_dx1dx3 = A130*((v_p1(j+1,1) - v_m1(j+1,1) - v_p1(j,1) + v_m1(j,1))/(4*dx*dz));
      dV1_dx2dx1 = A210*((v_p1(j+1,1) - v_m1(j+1,1) - v_p1(j,1) + v_m1(j,1))/(4*dy*dx));
      dV1_dx2dx2 = A220*((v(j+2,1) - 2*v(j+1,1) + v(j,1))/(dy^2));
      dV1_dx2dx3 = A230*((v_p1(j+1,1) - v_m1(j+1,1) - v_p1(j,1) + v_m1(j,1))/(4*dy*dz));
      dV1_dx3dx1 = A310*((v_p1(j+1,1) - v_m1(j+1,1) - v_p1(j,1) + v_m1(j,1))/(4*dz*dx));
      dV1_dx3dx2 = A320*((v_p1(j+1,1) - v_m1(j+1,1) - v_p1(j,1) + v_m1(j,1))/(4*dz*dy));
      dV1_dx3dx3 = A330*((v_m1(j,1) - 2*v(j,1) + v_p1(j,1))/(dz^2));
   end

   if(v_p1(j,2) ~= 0) && (v_p1(j+1,2) ~= 0)
      dV2_dx1dx1 = A110*((v_m1(j,2) - 2*v(j,2) + v_p1(j,2))/(dx^2));
      dV2_dx1dx2 = A120*((v_p1(j+1,2) - v_m1(j+1,2) - v_p1(j,2) + v_m1(j,2))/(4*dx*dy));
      dV2_dx1dx3 = A130*((v_p1(j+1,2) - v_m1(j+1,2) - v_p1(j,2) + v_m1(j,2))/(4*dx*dz));
      dV2_dx2dx1 = A210*((v_p1(j+1,2) - v_m1(j+1,2) - v_p1(j,2) + v_m1(j,2))/(4*dy*dx));
      dV2_dx2dx2 = A220*((v(j+2,2) - 2*v(j+1,2) + v(j,2))/(dy^2));
      dV2_dx2dx3 = A230*((v_p1(j+1,2) - v_m1(j+1,2) - v_p1(j,2) + v_m1(j,2))/(4*dy*dz));
      dV2_dx3dx1 = A310*((v_p1(j+1,2) - v_m1(j+1,2) - v_p1(j,2) + v_m1(j,2))/(4*dz*dx));
      dV2_dx3dx2 = A320*((v_p1(j+1,2) - v_m1(j+1,2) - v_p1(j,2) + v_m1(j,2))/(4*dz*dy));
      dV2_dx3dx3 = A330*((v_m1(j,2) - 2*v(j,2) + v_p1(j,2))/(dz^2));
   end

   if(v_p1(j,3) ~= 0) && (v_p1(j+1,3) ~= 0)
      dV3_dx1dx1 = A110*((v_m1(j,3) - 2*v(j,3) + v_p1(j,3))/(dx^2));
      dV3_dx1dx2 = A120*((v_p1(j+1,3) - v_m1(j+1,3) - v_p1(j,3) + v_m1(j,3))/(4*dx*dy));
      dV3_dx1dx3 = A130*((v_p1(j+1,3) - v_m1(j+1,3) - v_p1(j,3) + v_m1(j,3))/(4*dx*dz));
      dV3_dx2dx1 = A210*((v_p1(j+1,3) - v_m1(j+1,3) - v_p1(j,3) + v_m1(j,3))/(4*dy*dx));
      dV3_dx2dx2 = A220*((v(j+2,3) - 2*v(j+1,3) + v(j,3))/(dy^2));
      dV3_dx2dx3 = A230*((v_p1(j+1,3) - v_m1(j+1,3) - v_p1(j,3) + v_m1(j,3))/(4*dy*dz));
      dV3_dx3dx1 = A310*((v_p1(j+1,3) - v_m1(j+1,3) - v_p1(j,3) + v_m1(j,3))/(4*dz*dx));
      dV3_dx3dx2 = A320*((v_p1(j+1,3) - v_m1(j+1,3) - v_p1(j,3) + v_m1(j,3))/(4*dz*dy));
      dV3_dx3dx3 = A330*((v_m1(j,3) - 2*v(j,3) + v_p1(j,3))/(dz^2));
   end

end

if (j > 1 && j < 31)
   % Here we are estimating the numerical derivatives away from the wall
   % so no boundary conditions are imposed in the code.

   if (v_p1(j+1,1) ~= 0) && (v_p1(j,1) ~= 0) && (v_p1(j-1,1) ~= 0)
      dV1_dx1dx1 = A110*((v_m1(j,1) - 2*v(j,1) + v_p1(j,1))/(dx^2));
```

```
    dV1_dx1dx2 = A120*((v_p1(j+1,1) - v_m1(j+1,1) - v_p1(j-1,1) + v_m1(j-1,1))/(4*dx*dy));
    dV1_dx1dx3 = A130*((v_p1(j+1,1) - v_m1(j+1,1) - v_p1(j-1,1) + v_m1(j-1,1))/(4*dx*dz));
    dV1_dx2dx1 = A210*((v_p1(j+1,1) - v_m1(j+1,1) - v_p1(j-1,1) + v_m1(j-1,1))/(4*dy*dx));
    dV1_dx2dx2 = A220*((v(j+1,1) - 2*v(j,1) + v(j-1,1))/(dy^2));
    dV1_dx2dx3 = A230*((v_p1(j+1,1) - v_m1(j+1,1) - v_p1(j-1,1) + v_m1(j-1,1))/(4*dy*dz));
    dV1_dx3dx1 = A310*((v_p1(j+1,1) - v_m1(j+1,1) - v_p1(j-1,1) + v_m1(j-1,1))/(4*dz*dx));
    dV1_dx3dx2 = A320*((v_p1(j+1,1) - v_m1(j+1,1) - v_p1(j-1,1) + v_m1(j-1,1))/(4*dz*dy));
    dV1_dx3dx3 = A330*((v_m1(j,1) - 2*v(j,1) + v_p1(j,1))/(dz^2));
  end

  if (v_p1(j+1,2) ~= 0) && (v_p1(j,2) ~= 0) && (v_p1(j-1,2) ~= 0)
    dV2_dx1dx1 = A110*((v_m1(j,2) - 2*v(j,2) + v_p1(j,2))/(dx^2));
    dV2_dx1dx2 = A120*((v_p1(j+1,2) - v_m1(j+1,2) - v_p1(j-1,2) + v_m1(j-1,2))/(4*dx*dy));
    dV2_dx1dx3 = A130*((v_p1(j+1,2) - v_m1(j+1,2) - v_p1(j-1,2) + v_m1(j-1,2))/(4*dx*dz));
    dV2_dx2dx1 = A210*((v_p1(j+1,2) - v_m1(j+1,2) - v_p1(j-1,2) + v_m1(j-1,2))/(4*dy*dx));
    dV2_dx2dx2 = A220*((v(j+1,2) - 2*v(j,2) + v(j-1,2))/(dy^2));
    dV2_dx2dx3 = A230*((v_p1(j+1,2) - v_m1(j+1,2) - v_p1(j-1,2) + v_m1(j-1,2))/(4*dy*dz));
    dV2_dx3dx1 = A310*((v_p1(j+1,2) - v_m1(j+1,2) - v_p1(j-1,2) + v_m1(j-1,2))/(4*dz*dx));
    dV2_dx3dx2 = A320*((v_p1(j+1,2) - v_m1(j+1,2) - v_p1(j-1,2) + v_m1(j-1,2))/(4*dz*dy));
    dV2_dx3dx3 = A330*((v_m1(j,2) - 2*v(j,2) + v_p1(j,2))/(dz^2));
  end

  if (v_p1(j+1,3) ~= 0) && (v_p1(j,3) ~= 0) && (v_p1(j-1,3) ~= 0)
    dV3_dx1dx1 = A110*((v_m1(j,3) - 2*v(j,3) + v_p1(j,3))/(dx^2));
    dV3_dx1dx2 = A120*((v_p1(j+1,3) - v_m1(j+1,3) - v_p1(j-1,3) + v_m1(j-1,3))/(4*dx*dy));
    dV3_dx1dx3 = A130*((v_p1(j+1,3) - v_m1(j+1,3) - v_p1(j-1,3) + v_m1(j-1,3))/(4*dx*dz));
    dV3_dx2dx1 = A210*((v_p1(j+1,3) - v_m1(j+1,3) - v_p1(j-1,3) + v_m1(j-1,3))/(4*dy*dx));
    dV3_dx2dx2 = A220*((v(j+1,3) - 2*v(j,3) + v(j-1,3))/(dy^2));
    dV3_dx2dx3 = A230*((v_p1(j+1,3) - v_m1(j+1,3) - v_p1(j-1,3) + v_m1(j-1,3))/(4*dy*dz));
    dV3_dx3dx1 = A310*((v_p1(j+1,3) - v_m1(j+1,3) - v_p1(j-1,3) + v_m1(j-1,3))/(4*dz*dx));
    dV3_dx3dx2 = A320*((v_p1(j+1,3) - v_m1(j+1,3) - v_p1(j-1,3) + v_m1(j-1,3))/(4*dz*dy));
    dV3_dx3dx3 = A330*((v_m1(j,3) - 2*v(j,3) + v_p1(j,3))/(dz^2));
  end

end

if (j == 31)
%    Here we again impose the no slip boundary condition, but instead we
%    equate the j+1 term equal to zero.

  if (v_p1(j,1) ~= 0) && (v(j,1) ~= 0) && (v_m1(j,1) ~= 0)
    dV1_dx1dx1 = A110*((v_m1(j,1) - 2*v(j,1) + v_p1(j,1))/(dx^2));
    dV1_dx1dx2 = A120*((v_p1(j,1) - v_m1(j,1) - v_p1(j-1,1) + v_m1(j-1,1))/(4*dx*dy));
    dV1_dx1dx3 = A130*((v_p1(j,1) - v_m1(j,1) - v_p1(j-1,1) + v_m1(j-1,1))/(4*dx*dz));
    dV1_dx2dx1 = A210*((v_p1(j,1) - v_m1(j,1) - v_p1(j-1,1) + v_m1(j-1,1))/(4*dy*dx));
    dV1_dx2dx2 = A220*((v(j,1) - 2*v(j-1,1) + v(j-2,1))/(dy^2));
    dV1_dx2dx3 = A230*((v_p1(j,1) - v_m1(j,1) - v_p1(j-1,1) + v_m1(j-1,1))/(4*dy*dz));
    dV1_dx3dx1 = A310*((v_p1(j,1) - v_m1(j,1) - v_p1(j-1,1) + v_m1(j-1,1))/(4*dz*dx));
    dV1_dx3dx2 = A320*((v_p1(j,1) - v_m1(j,1) - v_p1(j-1,1) + v_m1(j-1,1))/(4*dz*dy));
    dV1_dx3dx3 = A330*((v_m1(j,1) - 2*v(j,1) + v_p1(j,1))/(dz^2));
  end

  if (v_p1(j,2) ~= 0) && (v_p1(j-1,2) ~= 0)
    dV2_dx1dx1 = A110*((v_m1(j,2) - 2*v(j,2) + v_p1(j,2))/(dx^2));
    dV2_dx1dx2 = A120*((v_p1(j,2) - v_m1(j,2) - v_p1(j-1,2) + v_m1(j-1,2))/(4*dx*dy));
    dV2_dx1dx3 = A130*((v_p1(j,2) - v_m1(j,2) - v_p1(j-1,2) + v_m1(j-1,2))/(4*dx*dz));
    dV2_dx2dx1 = A210*((v_p1(j,2) - v_m1(j,2) - v_p1(j-1,2) + v_m1(j-1,2))/(4*dy*dx));
    dV2_dx2dx2 = A220*((v(j,2) - 2*v(j-1,2) + v(j-2,2))/(dy^2));
    dV2_dx2dx3 = A230*((v_p1(j,2) - v_m1(j,2) - v_p1(j-1,2) + v_m1(j-1,2))/(4*dy*dz));
    dV2_dx3dx1 = A310*((v_p1(j,2) - v_m1(j,2) - v_p1(j-1,2) + v_m1(j-1,2))/(4*dz*dx));
```

```
    dV2_dx3dx2 = A320*((v_p1(j,2) - v_m1(j,2) - v_p1(j-1,2) + v_m1(j-1,2))/(4*dz*dy));
    dV2_dx3dx3 = A330*((v_m1(j,2) - 2*v(j,2) + v_p1(j,2))/(dz^2));
  end

  if (v_p1(j,3) ~= 0) && (v_p1(j-1,3) ~= 0)
    dV3_dx1dx1 = A110*((v_m1(j,3) - 2*v(j,3) + v_p1(j,3))/(dx^2));
    dV3_dx1dx2 = A120*((v_p1(j,3) - v_m1(j,3) - v_p1(j-1,3) + v_m1(j-1,3))/(4*dx*dy));
    dV3_dx1dx3 = A130*((v_p1(j,3) - v_m1(j,3) - v_p1(j-1,3) + v_m1(j-1,3))/(4*dx*dz));
    dV3_dx2dx1 = A210*((v_p1(j,3) - v_m1(j,3) - v_p1(j-1,3) + v_m1(j-1,3))/(4*dy*dx));
    dV3_dx2dx2 = A220*((v(j,3) - 2*v(j-1,3) + v(j-2,3))/(dy^2));
    dV3_dx2dx3 = A230*((v_p1(j,3) - v_m1(j,3) - v_p1(j-1,3) + v_m1(j-1,3))/(4*dy*dz));
    dV3_dx3dx1 = A310*((v_p1(j,3) - v_m1(j,3) - v_p1(j-1,3) + v_m1(j-1,3))/(4*dz*dx));
    dV3_dx3dx2 = A320*((v_p1(j,3) - v_m1(j,3) - v_p1(j-1,3) + v_m1(j-1,3))/(4*dz*dy));
    dV3_dx3dx3 = A330*((v_m1(j,3) - 2*v(j,3) + v_p1(j,3))/(dz^2));
  end

end

d2V_dxidxj(:,1) = [dV1_dx1dx1 dV1_dx1dx2 dV1_dx1dx3 dV1_dx2dx1 dV1_dx2dx2 dV1_dx2dx3 dV1_dx3dx1 dV1_dx3dx2
dV1_dx3dx3];

d2V_dxidxj(:,2) = [dV2_dx1dx1 dV2_dx1dx2 dV2_dx1dx3 dV2_dx2dx1 dV2_dx2dx2 dV2_dx2dx3 dV2_dx3dx1 dV2_dx3dx2
dV2_dx3dx3];

d2V_dxidxj(:,3) = [dV3_dx1dx1 dV3_dx1dx2 dV3_dx1dx3 dV3_dx2dx1 dV3_dx2dx2 dV3_dx2dx3 dV3_dx3dx1 dV3_dx3dx2
dV3_dx3dx3];

d2V_dxidxj(isnan(d2V_dxidxj)) = 0;
d2V_dxidxj(isinf(d2V_dxidxj)) = 0;

m = sum(d2V_dxidxj(:,k));
end
```

E.3.4 LddD.m

```
function [LddD] = LddD(D,A)

% Function written to compute the 4th order L tensor double dotted into the
% 2nd order rate of strain tensor (D).

[eigAvec,eigAval] = eig(reshape(A,3,3));   % Calculation of the numerical eigenvalues and eigenvectors.

eigAvec = fliplr(eigAvec);
eigAval = rot90(rot90(eigAval));

% Assigning the numerical values to discrete variables to ease in
% calculation.

eigAval1 = eigAval(1);
eigAval2 = eigAval(5);
eigAval3 = eigAval(9);

eigAvec1 = eigAvec(:,1);
eigAvec2 = eigAvec(:,2);
eigAvec3 = eigAvec(:,3);

%% CALCULATION FOR I = 1
```

```
% Calculation of the 1,1 Component of the LddD 2nd Order Tensor

LddD1111 = eigAvec1(1)^4                          *D(1);
LddD1112 = eigAvec1(1)^3 *eigAvec1(2)             *D(4);
LddD1113 = eigAvec1(1)^3           *eigAvec1(3)   *D(7);
LddD1121 = eigAvec1(1)^3 *eigAvec1(2)             *D(2);
LddD1122 = eigAvec1(1)^2 *eigAvec1(2)^2           *D(5);
LddD1123 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)   *D(8);
LddD1131 = eigAvec1(1)^3           *eigAvec1(3)   *D(3);
LddD1132 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)   *D(6);
LddD1133 = eigAvec1(1)^2           *eigAvec1(3)^2 *D(9);

LddD11_e1 = eigAval1*(LddD1111 + LddD1112 + LddD1113 + LddD1121 + LddD1122 + LddD1123 + LddD1131 +
LddD1132 + LddD1133);

% Calculation of the 1,2 Component of the LddD 2nd Order Tensor

LddD1211 = eigAvec1(1)^3 *eigAvec1(2)             *D(1);
LddD1212 = eigAvec1(1)^2 *eigAvec1(2)^2           *D(4);
LddD1213 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)   *D(7);
LddD1221 = eigAvec1(1)^2 *eigAvec1(2)^2           *D(2);
LddD1222 = eigAvec1(1)   *eigAvec1(2)^3           *D(5);
LddD1223 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)   *D(8);
LddD1231 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)   *D(3);
LddD1232 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)   *D(6);
LddD1233 = eigAvec1(1)   *eigAvec1(2)  *eigAvec1(3)^2 *D(9);

LddD12_e1 = eigAval1*(LddD1211 + LddD1212 + LddD1213 + LddD1221 + LddD1222 + LddD1223 + LddD1231 +
LddD1232 + LddD1233);

% Calculation of the 1,3 Component of the LddD 2nd Order Tensor

LddD1311 = eigAvec1(1)^3           *eigAvec1(3)    *D(1);
LddD1312 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)    *D(4);
LddD1313 = eigAvec1(1)^2           *eigAvec1(3)^2  *D(7);
LddD1321 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)    *D(2);
LddD1322 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)   *D(5);
LddD1323 = eigAvec1(1)   *eigAvec1(2)  *eigAvec1(3)^2  *D(8);
LddD1331 = eigAvec1(1)^2           *eigAvec1(3)^2  *D(3);
LddD1332 = eigAvec1(1)   *eigAvec1(2)  *eigAvec1(3)^2  *D(6);
LddD1333 = eigAvec1(1)             *eigAvec1(3)^3  *D(9);

LddD13_e1 = eigAval1*(LddD1311 + LddD1312 + LddD1313 + LddD1321 + LddD1322 + LddD1323 + LddD1331 +
LddD1332 + LddD1333);

% Calculation of the 2,1 Component of the LddD 2nd Order Tensor

LddD21_e1 = LddD12_e1;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the LddD 2nd Order Tensor

LddD2211 = eigAvec1(1)^2 *eigAvec1(2)^2           *D(1);
LddD2212 = eigAvec1(1)   *eigAvec1(2)^3           *D(4);
LddD2213 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)   *D(7);
LddD2221 = eigAvec1(1)   *eigAvec1(2)^3           *D(2);
LddD2222 =               eigAvec1(2)^4            *D(5);
```

```
LddD2223 =              eigAvec1(2)^3 *eigAvec1(3)     *D(8);
LddD2231 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)     *D(3);
LddD2232 =              eigAvec1(2)^3 *eigAvec1(3)     *D(6);
LddD2233 =              eigAvec1(2)^2 *eigAvec1(3)^2   *D(9);

LddD22_e1 = eigAval1*(LddD2211 + LddD2212 + LddD2213 + LddD2221 + LddD2222 + LddD2223 + LddD2231 +
LddD2232 + LddD2233);

% Calculation of the 2,3 Component of the LddD 2nd Order Tensor

LddD2311 = eigAvec1(1)^2 *eigAvec1(2)   *eigAvec1(3)     *D(1);
LddD2312 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)     *D(4);
LddD2313 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *D(7);
LddD2321 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)     *D(2);
LddD2322 =              eigAvec1(2)^3 *eigAvec1(3)     *D(5);
LddD2323 =              eigAvec1(2)^2 *eigAvec1(3)^2   *D(8);
LddD2331 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *D(3);
LddD2332 =              eigAvec1(2)^2 *eigAvec1(3)^2   *D(6);
LddD2333 =              eigAvec1(2)   *eigAvec1(3)^3   *D(9);

LddD23_e1 = eigAval1*(LddD2311 + LddD2312 + LddD2313 + LddD2321 + LddD2322 + LddD2323 + LddD2331 +
LddD2332 + LddD2333);

% Calculation of the 3,1 Component of the LddD 2nd Order Tensor

LddD31_e1 = LddD13_e1;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the LddD 2nd Order Tensor

LddD32_e1 = LddD23_e1;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the LddD 2nd Order Tensor

LddD3311 = eigAvec1(1)^2              *eigAvec1(3)^2   *D(1);
LddD3312 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *D(4);
LddD3313 = eigAvec1(1)                *eigAvec1(3)^3   *D(7);
LddD3321 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *D(2);
LddD3322 =              eigAvec1(2)^2 *eigAvec1(3)^2   *D(5);
LddD3323 =              eigAvec1(2)   *eigAvec1(3)^3   *D(8);
LddD3331 = eigAvec1(1)                *eigAvec1(3)^3   *D(3);
LddD3332 =              eigAvec1(2)   *eigAvec1(3)^3   *D(6);
LddD3333 =                          eigAvec1(3)^4   *D(9);

LddD33_e1 = eigAval1*(LddD3311 + LddD3312 + LddD3313 + LddD3321 + LddD3322 + LddD3323 + LddD3331 +
LddD3332 + LddD3333);

%% CALCULATION FOR I = 2

% Calculation of the 1,1 Component of the LddD 2nd Order Tensor

LddD1111 = eigAvec2(1)^4                          *D(1);
LddD1112 = eigAvec2(1)^3 *eigAvec2(2)                *D(4);
LddD1113 = eigAvec2(1)^3              *eigAvec2(3)    *D(7);
LddD1121 = eigAvec2(1)^3 *eigAvec2(2)                *D(2);
LddD1122 = eigAvec2(1)^2 *eigAvec2(2)^2              *D(5);
LddD1123 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *D(8);
LddD1131 = eigAvec2(1)^3              *eigAvec2(3)    *D(3);
```

```
LddD1132 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *D(6);
LddD1133 = eigAvec2(1)^2                *eigAvec2(3)^2  *D(9);

LddD11_e2 = eigAval2*(LddD1111 + LddD1112 + LddD1113 + LddD1121 + LddD1122 + LddD1123 + LddD1131 +
LddD1132 + LddD1133);

% Calculation of the 1,2 Component of the LddD 2nd Order Tensor

LddD1211 = eigAvec2(1)^3 *eigAvec2(2)                *D(1);
LddD1212 = eigAvec2(1)^2 *eigAvec2(2)^2              *D(4);
LddD1213 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *D(7);
LddD1221 = eigAvec2(1)^2 *eigAvec2(2)^2              *D(2);
LddD1222 = eigAvec2(1)   *eigAvec2(2)^3              *D(5);
LddD1223 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *D(8);
LddD1231 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *D(3);
LddD1232 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *D(6);
LddD1233 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *D(9);

LddD12_e2 = eigAval2*(LddD1211 + LddD1212 + LddD1213 + LddD1221 + LddD1222 + LddD1223 + LddD1231 +
LddD1232 + LddD1233);

% Calculation of the 1,3 Component of the LddD 2nd Order Tensor

LddD1311 = eigAvec2(1)^3             *eigAvec2(3)     *D(1);
LddD1312 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)     *D(4);
LddD1313 = eigAvec2(1)^2             *eigAvec2(3)^2   *D(7);
LddD1321 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)     *D(2);
LddD1322 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)     *D(5);
LddD1323 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2   *D(8);
LddD1331 = eigAvec2(1)^2             *eigAvec2(3)^2   *D(3);
LddD1332 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2   *D(6);
LddD1333 = eigAvec2(1)               *eigAvec2(3)^3   *D(9);

LddD13_e2 = eigAval2*(LddD1311 + LddD1312 + LddD1313 + LddD1321 + LddD1322 + LddD1323 + LddD1331 +
LddD1332 + LddD1333);

% Calculation of the 2,1 Component of the LddD 2nd Order Tensor

LddD21_e2 = LddD12_e2;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the LddD 2nd Order Tensor

LddD2211 = eigAvec2(1)^2 *eigAvec2(2)^2              *D(1);
LddD2212 = eigAvec2(1)   *eigAvec2(2)^3              *D(4);
LddD2213 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)     *D(7);
LddD2221 = eigAvec2(1)   *eigAvec2(2)^3              *D(2);
LddD2222 =               eigAvec2(2)^4               *D(5);
LddD2223 =               eigAvec2(2)^3 *eigAvec2(3)     *D(8);
LddD2231 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)     *D(3);
LddD2232 =               eigAvec2(2)^3 *eigAvec2(3)     *D(6);
LddD2233 =               eigAvec2(2)^2 *eigAvec2(3)^2   *D(9);

LddD22_e2 = eigAval2*(LddD2211 + LddD2212 + LddD2213 + LddD2221 + LddD2222 + LddD2223 + LddD2231 +
LddD2232 + LddD2233);

% Calculation of the 2,3 Component of the LddD 2nd Order Tensor
```

```
LddD2311 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *D(1);
LddD2312 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *D(4);
LddD2313 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *D(7);
LddD2321 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *D(2);
LddD2322 =               eigAvec2(2)^3 *eigAvec2(3)     *D(5);
LddD2323 =               eigAvec2(2)^2 *eigAvec2(3)^2   *D(8);
LddD2331 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *D(3);
LddD2332 =               eigAvec2(2)^2 *eigAvec2(3)^2   *D(6);
LddD2333 =               eigAvec2(2)   *eigAvec2(3)^3   *D(9);

LddD23_e2 = eigAval2*(LddD2311 + LddD2312 + LddD2313 + LddD2321 + LddD2322 + LddD2323 + LddD2331 +
LddD2332 + LddD2333);

% Calculation of the 3,1 Component of the LddD 2nd Order Tensor

LddD31_e2 = LddD13_e2;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the LddD 2nd Order Tensor

LddD32_e2 = LddD23_e2;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the LddD 2nd Order Tensor

LddD3311 = eigAvec2(1)^2              *eigAvec2(3)^2   *D(1);
LddD3312 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2   *D(4);
LddD3313 = eigAvec2(1)                *eigAvec2(3)^3   *D(7);
LddD3321 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2   *D(2);
LddD3322 =               eigAvec2(2)^2 *eigAvec2(3)^2   *D(5);
LddD3323 =               eigAvec2(2)   *eigAvec2(3)^3   *D(8);
LddD3331 = eigAvec2(1)                *eigAvec2(3)^3   *D(3);
LddD3332 =               eigAvec2(2)   *eigAvec2(3)^3   *D(6);
LddD3333 =                             eigAvec2(3)^4   *D(9);

LddD33_e2 = eigAval2*(LddD3311 + LddD3312 + LddD3313 + LddD3321 + LddD3322 + LddD3323 + LddD3331 +
LddD3332 + LddD3333);

%% CALCULATION FOR I = 3

% Calculation of the 1,1 Component of the LddD 2nd Order Tensor

LddD1111 = eigAvec3(1)^4                          *D(1);
LddD1112 = eigAvec3(1)^3 *eigAvec3(2)             *D(4);
LddD1113 = eigAvec3(1)^3             *eigAvec3(3) *D(7);
LddD1121 = eigAvec3(1)^3 *eigAvec3(2)             *D(2);
LddD1122 = eigAvec3(1)^2 *eigAvec3(2)^2           *D(5);
LddD1123 = eigAvec3(1)^2 *eigAvec3(2)  *eigAvec3(3) *D(8);
LddD1131 = eigAvec3(1)^3             *eigAvec3(3) *D(3);
LddD1132 = eigAvec3(1)^2 *eigAvec3(2)  *eigAvec3(3) *D(6);
LddD1133 = eigAvec3(1)^2             *eigAvec3(3)^2  *D(9);

LddD11_e3 = eigAval3*(LddD1111 + LddD1112 + LddD1113 + LddD1121 + LddD1122 + LddD1123 + LddD1131 +
LddD1132 + LddD1133);

% Calculation of the 1,2 Component of the LddD 2nd Order Tensor

LddD1211 = eigAvec3(1)^3 *eigAvec3(2)             *D(1);
LddD1212 = eigAvec3(1)^2 *eigAvec3(2)^2           *D(4);
```

```
LddD1213 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *D(7);
LddD1221 = eigAvec3(1)^2 *eigAvec3(2)^2                 *D(2);
LddD1222 = eigAvec3(1)   *eigAvec3(2)^3                 *D(5);
LddD1223 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *D(8);
LddD1231 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *D(3);
LddD1232 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *D(6);
LddD1233 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2  *D(9);

LddD12_e3 = eigAval3*(LddD1211 + LddD1212 + LddD1213 + LddD1221 + LddD1222 + LddD1223 + LddD1231 +
LddD1232 + LddD1233);

% Calculation of the 1,3 Component of the LddD 2nd Order Tensor

LddD1311 = eigAvec3(1)^3                *eigAvec3(3)     *D(1);
LddD1312 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)     *D(4);
LddD1313 = eigAvec3(1)^2                *eigAvec3(3)^2   *D(7);
LddD1321 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)     *D(2);
LddD1322 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)     *D(5);
LddD1323 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *D(8);
LddD1331 = eigAvec3(1)^2                *eigAvec3(3)^2   *D(3);
LddD1332 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *D(6);
LddD1333 = eigAvec3(1)                  *eigAvec3(3)^3   *D(9);

LddD13_e3 = eigAval3*(LddD1311 + LddD1312 + LddD1313 + LddD1321 + LddD1322 + LddD1323 + LddD1331 +
LddD1332 + LddD1333);

% Calculation of the 2,1 Component of the LddD 2nd Order Tensor

LddD21_e3 = LddD12_e3;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the LddD 2nd Order Tensor

LddD2211 = eigAvec3(1)^2 *eigAvec3(2)^2                 *D(1);
LddD2212 = eigAvec3(1)   *eigAvec3(2)^3                 *D(4);
LddD2213 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *D(7);
LddD2221 = eigAvec3(1)   *eigAvec3(2)^3                 *D(2);
LddD2222 =               eigAvec3(2)^4                  *D(5);
LddD2223 =               eigAvec3(2)^3 *eigAvec3(3)     *D(8);
LddD2231 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *D(3);
LddD2232 =               eigAvec3(2)^3 *eigAvec3(3)     *D(6);
LddD2233 =               eigAvec3(2)^2 *eigAvec3(3)^2   *D(9);

LddD22_e3 = eigAval3*(LddD2211 + LddD2212 + LddD2213 + LddD2221 + LddD2222 + LddD2223 + LddD2231 +
LddD2232 + LddD2233);

% Calculation of the 2,3 Component of the LddD 2nd Order Tensor

LddD2311 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)     *D(1);
LddD2312 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)     *D(4);
LddD2313 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *D(7);
LddD2321 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)     *D(2);
LddD2322 =               eigAvec3(2)^3 *eigAvec3(3)     *D(5);
LddD2323 =               eigAvec3(2)^2 *eigAvec3(3)^2   *D(8);
LddD2331 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *D(3);
LddD2332 =               eigAvec3(2)^2 *eigAvec3(3)^2   *D(6);
LddD2333 =               eigAvec3(2)   *eigAvec3(3)^3   *D(9);
```

```
LddD23_e3 = eigAval3*(LddD2311 + LddD2312 + LddD2313 + LddD2321 + LddD2322 + LddD2323 + LddD2331 +
LddD2332 + LddD2333);

% Calculation of the 3,1 Component of the LddD 2nd Order Tensor

LddD31_e3 = LddD13_e3;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the LddD 2nd Order Tensor

LddD32_e3 = LddD23_e3;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the LddD 2nd Order Tensor

LddD3311 = eigAvec3(1)^2              *eigAvec3(3)^2   *D(1);
LddD3312 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *D(4);
LddD3313 = eigAvec3(1)              *eigAvec3(3)^3   *D(7);
LddD3321 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *D(2);
LddD3322 =            eigAvec3(2)^2 *eigAvec3(3)^2   *D(5);
LddD3323 =            eigAvec3(2)   *eigAvec3(3)^3   *D(8);
LddD3331 = eigAvec3(1)              *eigAvec3(3)^3   *D(3);
LddD3332 =            eigAvec3(2)   *eigAvec3(3)^3   *D(6);
LddD3333 =                        eigAvec3(3)^4   *D(9);

LddD33_e3 = eigAval2*(LddD3311 + LddD3312 + LddD3313 + LddD3321 + LddD3322 + LddD3323 + LddD3331 +
LddD3332 + LddD3333);

LddD11 = LddD11_e1 + LddD11_e2 + LddD11_e3;
LddD12 = LddD12_e1 + LddD12_e2 + LddD12_e3;
LddD13 = LddD13_e1 + LddD13_e2 + LddD13_e3;
LddD21 = LddD21_e1 + LddD21_e2 + LddD21_e3;
LddD22 = LddD22_e1 + LddD22_e2 + LddD22_e3;
LddD23 = LddD23_e1 + LddD23_e2 + LddD23_e3;
LddD31 = LddD32_e1 + LddD31_e2 + LddD31_e3;
LddD32 = LddD32_e1 + LddD32_e2 + LddD32_e3;
LddD33 = LddD33_e1 + LddD33_e2 + LddD33_e3;

LddD = [LddD11 LddD12 LddD13 LddD21 LddD22 LddD23 LddD31 LddD32 LddD33];

LddD(isnan(LddD)) = 0;
LddD(isinf(LddD)) = 0;

end
```

E.3.5 LddC.m

```
function [LddC] = LddC(C,A)

% Function written to compute the 4th order L tensor double dotted into the
% 2nd order anisotropy tensor (C).

[eigAvec,eigAval] = eig(reshape(A,3,3));   % Calculation of the numerical eigenvalues and eigenvectors.

eigAvec = fliplr(eigAvec);
eigAval = rot90(rot90(eigAval));

% Assigning the numerical values to discrete variables to ease in
% calculation.
```

```
eigAval1 = eigAval(1);
eigAval2 = eigAval(5);
eigAval3 = eigAval(9);

eigAvec1 = eigAvec(:,1);
eigAvec2 = eigAvec(:,2);
eigAvec3 = eigAvec(:,3);

%% CALCULATION FOR I = 1

% Calculation of the 1,1 Component of the LddC 2nd Order Tensor

LddC1111 = eigAvec1(1)^4                          *C(1);
LddC1112 = eigAvec1(1)^3 *eigAvec1(2)             *C(4);
LddC1113 = eigAvec1(1)^3             *eigAvec1(3) *C(7);
LddC1121 = eigAvec1(1)^3 *eigAvec1(2)             *C(2);
LddC1122 = eigAvec1(1)^2 *eigAvec1(2)^2           *C(5);
LddC1123 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3) *C(8);
LddC1131 = eigAvec1(1)^3             *eigAvec1(3) *C(3);
LddC1132 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3) *C(6);
LddC1133 = eigAvec1(1)^2             *eigAvec1(3)^2 *C(9);

LddC11_e1 = eigAval1*(LddC1111 + LddC1112 + LddC1113 + LddC1121 + LddC1122 + LddC1123 + LddC1131 + LddC1132
+ LddC1133);

% Calculation of the 1,2 Component of the LddC 2nd Order Tensor

LddC1211 = eigAvec1(1)^3 *eigAvec1(2)             *C(1);
LddC1212 = eigAvec1(1)^2 *eigAvec1(2)^2           *C(4);
LddC1213 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3) *C(7);
LddC1221 = eigAvec1(1)^2 *eigAvec1(2)^2           *C(2);
LddC1222 = eigAvec1(1)  *eigAvec1(2)^3            *C(5);
LddC1223 = eigAvec1(1)  *eigAvec1(2)^2 *eigAvec1(3) *C(8);
LddC1231 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3) *C(3);
LddC1232 = eigAvec1(1)  *eigAvec1(2)^2 *eigAvec1(3) *C(6);
LddC1233 = eigAvec1(1)  *eigAvec1(2)  *eigAvec1(3)^2 *C(9);

LddC12_e1 = eigAval1*(LddC1211 + LddC1212 + LddC1213 + LddC1221 + LddC1222 + LddC1223 + LddC1231 + LddC1232
+ LddC1233);

% Calculation of the 1,3 Component of the LddC 2nd Order Tensor

LddC1311 = eigAvec1(1)^3             *eigAvec1(3)  *C(1);
LddC1312 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)  *C(4);
LddC1313 = eigAvec1(1)^2             *eigAvec1(3)^2  *C(7);
LddC1321 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)  *C(2);
LddC1322 = eigAvec1(1)  *eigAvec1(2)^2 *eigAvec1(3)  *C(5);
LddC1323 = eigAvec1(1)  *eigAvec1(2)  *eigAvec1(3)^2  *C(8);
LddC1331 = eigAvec1(1)^2             *eigAvec1(3)^2  *C(3);
LddC1332 = eigAvec1(1)  *eigAvec1(2)  *eigAvec1(3)^2  *C(6);
LddC1333 = eigAvec1(1)               *eigAvec1(3)^3  *C(9);

LddC13_e1 = eigAval1*(LddC1311 + LddC1312 + LddC1313 + LddC1321 + LddC1322 + LddC1323 + LddC1331 + LddC1332
+ LddC1333);

% Calculation of the 2,1 Component of the LddC 2nd Order Tensor
```

```
LddC21_e1 = LddC12_e1;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the LddC 2nd Order Tensor

LddC2211 = eigAvec1(1)^2 *eigAvec1(2)^2                *C(1);
LddC2212 = eigAvec1(1)   *eigAvec1(2)^3                *C(4);
LddC2213 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)     *C(7);
LddC2221 = eigAvec1(1)   *eigAvec1(2)^3                *C(2);
LddC2222 =               eigAvec1(2)^4                 *C(5);
LddC2223 =               eigAvec1(2)^3 *eigAvec1(3)    *C(8);
LddC2231 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)     *C(3);
LddC2232 =               eigAvec1(2)^3 *eigAvec1(3)    *C(6);
LddC2233 =               eigAvec1(2)^2 *eigAvec1(3)^2   *C(9);

LddC22_e1 = eigAval1*(LddC2211 + LddC2212 + LddC2213 + LddC2221 + LddC2222 + LddC2223 + LddC2231 + LddC2232
+ LddC2233);

% Calculation of the 2,3 Component of the LddC 2nd Order Tensor

LddC2311 = eigAvec1(1)^2 *eigAvec1(2)   *eigAvec1(3)     *C(1);
LddC2312 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)     *C(4);
LddC2313 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *C(7);
LddC2321 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)     *C(2);
LddC2322 =               eigAvec1(2)^3 *eigAvec1(3)     *C(5);
LddC2323 =               eigAvec1(2)^2 *eigAvec1(3)^2   *C(8);
LddC2331 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *C(3);
LddC2332 =               eigAvec1(2)^2 *eigAvec1(3)^2   *C(6);
LddC2333 =               eigAvec1(2)   *eigAvec1(3)^3   *C(9);

LddC23_e1 = eigAval1*(LddC2311 + LddC2312 + LddC2313 + LddC2321 + LddC2322 + LddC2323 + LddC2331 + LddC2332
+ LddC2333);

% Calculation of the 3,1 Component of the LddC 2nd Order Tensor

LddC31_e1 = LddC13_e1;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the LddC 2nd Order Tensor

LddC32_e1 = LddC23_e1;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the LddC 2nd Order Tensor

LddC3311 = eigAvec1(1)^2               *eigAvec1(3)^2   *C(1);
LddC3312 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *C(4);
LddC3313 = eigAvec1(1)                  *eigAvec1(3)^3   *C(7);
LddC3321 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *C(2);
LddC3322 =               eigAvec1(2)^2 *eigAvec1(3)^2   *C(5);
LddC3323 =               eigAvec1(2)   *eigAvec1(3)^3   *C(8);
LddC3331 = eigAvec1(1)                  *eigAvec1(3)^3   *C(3);
LddC3332 =               eigAvec1(2)   *eigAvec1(3)^3   *C(6);
LddC3333 =                             eigAvec1(3)^4   *C(9);

LddC33_e1 = eigAval1*(LddC3311 + LddC3312 + LddC3313 + LddC3321 + LddC3322 + LddC3323 + LddC3331 + LddC3332
+ LddC3333);

%% CALCULATION FOR I = 2
```

```
% Calculation of the 1,1 Component of the LddC 2nd Order Tensor

LddC1111 = eigAvec2(1)^4                          *C(1);
LddC1112 = eigAvec2(1)^3 *eigAvec2(2)             *C(4);
LddC1113 = eigAvec2(1)^3           *eigAvec2(3)   *C(7);
LddC1121 = eigAvec2(1)^3 *eigAvec2(2)             *C(2);
LddC1122 = eigAvec2(1)^2 *eigAvec2(2)^2           *C(5);
LddC1123 = eigAvec2(1)^2 *eigAvec2(2)  *eigAvec2(3)   *C(8);
LddC1131 = eigAvec2(1)^3           *eigAvec2(3)   *C(3);
LddC1132 = eigAvec2(1)^2 *eigAvec2(2)  *eigAvec2(3)   *C(6);
LddC1133 = eigAvec2(1)^2           *eigAvec2(3)^2 *C(9);

LddC11_e2 = eigAval2*(LddC1111 + LddC1112 + LddC1113 + LddC1121 + LddC1122 + LddC1123 + LddC1131 + LddC1132
+ LddC1133);

% Calculation of the 1,2 Component of the LddC 2nd Order Tensor

LddC1211 = eigAvec2(1)^3 *eigAvec2(2)             *C(1);
LddC1212 = eigAvec2(1)^2 *eigAvec2(2)^2           *C(4);
LddC1213 = eigAvec2(1)^2 *eigAvec2(2)  *eigAvec2(3)   *C(7);
LddC1221 = eigAvec2(1)^2 *eigAvec2(2)^2           *C(2);
LddC1222 = eigAvec2(1)   *eigAvec2(2)^3           *C(5);
LddC1223 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)   *C(8);
LddC1231 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)   *C(3);
LddC1232 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)   *C(6);
LddC1233 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2 *C(9);

LddC12_e2 = eigAval2*(LddC1211 + LddC1212 + LddC1213 + LddC1221 + LddC1222 + LddC1223 + LddC1231 + LddC1232
+ LddC1233);

% Calculation of the 1,3 Component of the LddC 2nd Order Tensor

LddC1311 = eigAvec2(1)^3           *eigAvec2(3)   *C(1);
LddC1312 = eigAvec2(1)^2 *eigAvec2(2)  *eigAvec2(3)   *C(4);
LddC1313 = eigAvec2(1)^2           *eigAvec2(3)^2 *C(7);
LddC1321 = eigAvec2(1)^2 *eigAvec2(2)  *eigAvec2(3)   *C(2);
LddC1322 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)   *C(5);
LddC1323 = eigAvec2(1)   *eigAvec2(2)  *eigAvec2(3)^2 *C(8);
LddC1331 = eigAvec2(1)^2           *eigAvec2(3)^2 *C(3);
LddC1332 = eigAvec2(1)   *eigAvec2(2)  *eigAvec2(3)^2 *C(6);
LddC1333 = eigAvec2(1)             *eigAvec2(3)^3 *C(9);

LddC13_e2 = eigAval2*(LddC1311 + LddC1312 + LddC1313 + LddC1321 + LddC1322 + LddC1323 + LddC1331 + LddC1332
+ LddC1333);

% Calculation of the 2,1 Component of the LddC 2nd Order Tensor

LddC21_e2 = LddC12_e2;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the LddC 2nd Order Tensor

LddC2211 = eigAvec2(1)^2 *eigAvec2(2)^2           *C(1);
LddC2212 = eigAvec2(1)   *eigAvec2(2)^3           *C(4);
LddC2213 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)   *C(7);
LddC2221 = eigAvec2(1)   *eigAvec2(2)^3           *C(2);
LddC2222 =          eigAvec2(2)^4           *C(5);
```

```
LddC2223 =              eigAvec2(2)^3 *eigAvec2(3)     *C(8);
LddC2231 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)     *C(3);
LddC2232 =              eigAvec2(2)^3 *eigAvec2(3)     *C(6);
LddC2233 =              eigAvec2(2)^2 *eigAvec2(3)^2   *C(9);

LddC22_e2 = eigAval2*(LddC2211 + LddC2212 + LddC2213 + LddC2221 + LddC2222 + LddC2223 + LddC2231 + LddC2232
+ LddC2233);

% Calculation of the 2,3 Component of the LddC 2nd Order Tensor

LddC2311 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)     *C(1);
LddC2312 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)     *C(4);
LddC2313 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2   *C(7);
LddC2321 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)     *C(2);
LddC2322 =              eigAvec2(2)^3 *eigAvec2(3)     *C(5);
LddC2323 =              eigAvec2(2)^2 *eigAvec2(3)^2   *C(8);
LddC2331 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2   *C(3);
LddC2332 =              eigAvec2(2)^2 *eigAvec2(3)^2   *C(6);
LddC2333 =              eigAvec2(2)   *eigAvec2(3)^3   *C(9);

LddC23_e2 = eigAval2*(LddC2311 + LddC2312 + LddC2313 + LddC2321 + LddC2322 + LddC2323 + LddC2331 + LddC2332
+ LddC2333);

% Calculation of the 3,1 Component of the LddC 2nd Order Tensor

LddC31_e2 = LddC13_e2;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the LddC 2nd Order Tensor

LddC32_e2 = LddC23_e2;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the LddC 2nd Order Tensor

LddC3311 = eigAvec2(1)^2              *eigAvec2(3)^2   *C(1);
LddC3312 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2   *C(4);
LddC3313 = eigAvec2(1)              *eigAvec2(3)^3   *C(7);
LddC3321 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2   *C(2);
LddC3322 =              eigAvec2(2)^2 *eigAvec2(3)^2   *C(5);
LddC3323 =              eigAvec2(2)   *eigAvec2(3)^3   *C(8);
LddC3331 = eigAvec2(1)              *eigAvec2(3)^3   *C(3);
LddC3332 =              eigAvec2(2)   *eigAvec2(3)^3   *C(6);
LddC3333 =                        eigAvec2(3)^4   *C(9);

LddC33_e2 = eigAval2*(LddC3311 + LddC3312 + LddC3313 + LddC3321 + LddC3322 + LddC3323 + LddC3331 + LddC3332
+ LddC3333);

%% CALCULATION FOR I = 3

% Calculation of the 1,1 Component of the LddC 2nd Order Tensor

LddC1111 = eigAvec3(1)^4                          *C(1);
LddC1112 = eigAvec3(1)^3 *eigAvec3(2)               *C(4);
LddC1113 = eigAvec3(1)^3             *eigAvec3(3)   *C(7);
LddC1121 = eigAvec3(1)^3 *eigAvec3(2)               *C(2);
LddC1122 = eigAvec3(1)^2 *eigAvec3(2)^2             *C(5);
LddC1123 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)   *C(8);
LddC1131 = eigAvec3(1)^3             *eigAvec3(3)   *C(3);
```

```
LddC1132 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *C(6);
LddC1133 = eigAvec3(1)^2                *eigAvec3(3)^2  *C(9);

LddC11_e3 = eigAval3*(LddC1111 + LddC1112 + LddC1113 + LddC1121 + LddC1122 + LddC1123 + LddC1131 + LddC1132
+ LddC1133);

% Calculation of the 1,2 Component of the LddC 2nd Order Tensor

LddC1211 = eigAvec3(1)^3 *eigAvec3(2)                *C(1);
LddC1212 = eigAvec3(1)^2 *eigAvec3(2)^2              *C(4);
LddC1213 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *C(7);
LddC1221 = eigAvec3(1)^2 *eigAvec3(2)^2              *C(2);
LddC1222 = eigAvec3(1)   *eigAvec3(2)^3              *C(5);
LddC1223 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *C(8);
LddC1231 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *C(3);
LddC1232 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *C(6);
LddC1233 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2  *C(9);

LddC12_e3 = eigAval3*(LddC1211 + LddC1212 + LddC1213 + LddC1221 + LddC1222 + LddC1223 + LddC1231 + LddC1232
+ LddC1233);

% Calculation of the 1,3 Component of the LddC 2nd Order Tensor

LddC1311 = eigAvec3(1)^3             *eigAvec3(3)    *C(1);
LddC1312 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *C(4);
LddC1313 = eigAvec3(1)^2             *eigAvec3(3)^2  *C(7);
LddC1321 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *C(2);
LddC1322 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *C(5);
LddC1323 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2  *C(8);
LddC1331 = eigAvec3(1)^2             *eigAvec3(3)^2  *C(3);
LddC1332 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2  *C(6);
LddC1333 = eigAvec3(1)              *eigAvec3(3)^3  *C(9);

LddC13_e3 = eigAval3*(LddC1311 + LddC1312 + LddC1313 + LddC1321 + LddC1322 + LddC1323 + LddC1331 + LddC1332
+ LddC1333);

% Calculation of the 2,1 Component of the LddC 2nd Order Tensor

LddC21_e3 = LddC12_e3;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the LddC 2nd Order Tensor

LddC2211 = eigAvec3(1)^2 *eigAvec3(2)^2              *C(1);
LddC2212 = eigAvec3(1)   *eigAvec3(2)^3              *C(4);
LddC2213 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *C(7);
LddC2221 = eigAvec3(1)   *eigAvec3(2)^3              *C(2);
LddC2222 =               eigAvec3(2)^4               *C(5);
LddC2223 =               eigAvec3(2)^3 *eigAvec3(3)    *C(8);
LddC2231 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *C(3);
LddC2232 =               eigAvec3(2)^3 *eigAvec3(3)    *C(6);
LddC2233 =               eigAvec3(2)^2 *eigAvec3(3)^2  *C(9);

LddC22_e3 = eigAval3*(LddC2211 + LddC2212 + LddC2213 + LddC2221 + LddC2222 + LddC2223 + LddC2231 + LddC2232
+ LddC2233);

% Calculation of the 2,3 Component of the LddC 2nd Order Tensor
```

```
LddC2311 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)     *C(1);
LddC2312 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)     *C(4);
LddC2313 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *C(7);
LddC2321 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)     *C(2);
LddC2322 =               eigAvec3(2)^3 *eigAvec3(3)     *C(5);
LddC2323 =               eigAvec3(2)^2 *eigAvec3(3)^2   *C(8);
LddC2331 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *C(3);
LddC2332 =               eigAvec3(2)^2 *eigAvec3(3)^2   *C(6);
LddC2333 =               eigAvec3(2)   *eigAvec3(3)^3   *C(9);

LddC23_e3 = eigAval3*(LddC2311 + LddC2312 + LddC2313 + LddC2321 + LddC2322 + LddC2323 + LddC2331 + LddC2332
+ LddC2333);

% Calculation of the 3,1 Component of the LddC 2nd Order Tensor

LddC31_e3 = LddC13_e3;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the LddC 2nd Order Tensor

LddC32_e3 = LddC23_e3;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the LddC 2nd Order Tensor

LddC3311 = eigAvec3(1)^2                *eigAvec3(3)^2   *C(1);
LddC3312 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *C(4);
LddC3313 = eigAvec3(1)                  *eigAvec3(3)^3   *C(7);
LddC3321 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *C(2);
LddC3322 =               eigAvec3(2)^2 *eigAvec3(3)^2   *C(5);
LddC3323 =               eigAvec3(2)   *eigAvec3(3)^3   *C(8);
LddC3331 = eigAvec3(1)                  *eigAvec3(3)^3   *C(3);
LddC3332 =               eigAvec3(2)   *eigAvec3(3)^3   *C(6);
LddC3333 =                             eigAvec3(3)^4   *C(9);

LddC33_e3 = eigAval3*(LddC3311 + LddC3312 + LddC3313 + LddC3321 + LddC3322 + LddC3323 + LddC3331 + LddC3332
+ LddC3333);

%% Summing the terms over i (i.e. i = 1,2,3)

LddC11 = LddC11_e1 + LddC11_e2 + LddC11_e3;
LddC12 = LddC12_e1 + LddC12_e2 + LddC12_e3;
LddC13 = LddC13_e1 + LddC13_e2 + LddC13_e3;
LddC21 = LddC21_e1 + LddC21_e2 + LddC21_e3;
LddC22 = LddC22_e1 + LddC22_e2 + LddC22_e3;
LddC23 = LddC23_e1 + LddC23_e2 + LddC23_e3;
LddC31 = LddC32_e1 + LddC31_e2 + LddC31_e3;
LddC32 = LddC32_e1 + LddC32_e2 + LddC32_e3;
LddC33 = LddC33_e1 + LddC33_e2 + LddC33_e3;

%% Forming the L:C Tensor (2nd Order)

LddC = [LddC11 LddC12 LddC13 LddC21 LddC22 LddC23 LddC31 LddC32 LddC33];

end
```

E.3.6 MddC.m

```
function [MddC] = MddC(C,A)
```

```
% Function written to compute the 4th order M tensor double dotted into the
% 2nd order anistropy tensor (D).

[eigAvec,eigAval] = eig(reshape(A,3,3));   % Calculation of the numerical eigenvalues and eigenvectors.

eigAvec = fliplr(eigAvec);
eigAval = rot90(rot90(eigAval));

% Assigning the numerical values to discrete variables to ease in
% calculation.

eigAvec1 = eigAvec(:,1);
eigAvec2 = eigAvec(:,2);
eigAvec3 = eigAvec(:,3);

%% CALCULATION FOR I = 1

% Calculation of the 1,1 Component of the MddC 2nd Order Tensor

MddC1111 = eigAvec1(1)^4                          *C(1);
MddC1112 = eigAvec1(1)^3 *eigAvec1(2)             *C(4);
MddC1113 = eigAvec1(1)^3            *eigAvec1(3)  *C(7);
MddC1121 = eigAvec1(1)^3 *eigAvec1(2)             *C(2);
MddC1122 = eigAvec1(1)^2 *eigAvec1(2)^2           *C(5);
MddC1123 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)  *C(8);
MddC1131 = eigAvec1(1)^3            *eigAvec1(3)  *C(3);
MddC1132 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)  *C(6);
MddC1133 = eigAvec1(1)^2            *eigAvec1(3)^2  *C(9);

MddC11_e1 = (MddC1111 + MddC1112 + MddC1113 + MddC1121 + MddC1122 + MddC1123 + MddC1131 + MddC1132 +
MddC1133);

% Calculation of the 1,2 Component of the MddC 2nd Order Tensor

MddC1211 = eigAvec1(1)^3 *eigAvec1(2)             *C(1);
MddC1212 = eigAvec1(1)^2 *eigAvec1(2)^2           *C(4);
MddC1213 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)  *C(7);
MddC1221 = eigAvec1(1)^2 *eigAvec1(2)^2           *C(2);
MddC1222 = eigAvec1(1)  *eigAvec1(2)^3            *C(5);
MddC1223 = eigAvec1(1)  *eigAvec1(2)^2 *eigAvec1(3)  *C(8);
MddC1231 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)  *C(3);
MddC1232 = eigAvec1(1)  *eigAvec1(2)^2 *eigAvec1(3)  *C(6);
MddC1233 = eigAvec1(1)  *eigAvec1(2)  *eigAvec1(3)^2  *C(9);

MddC12_e1 = (MddC1211 + MddC1212 + MddC1213 + MddC1221 + MddC1222 + MddC1223 + MddC1231 + MddC1232 +
MddC1233);

% Calculation of the 1,3 Component of the MddC 2nd Order Tensor

MddC1311 = eigAvec1(1)^3            *eigAvec1(3)    *C(1);
MddC1312 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)    *C(4);
MddC1313 = eigAvec1(1)^2            *eigAvec1(3)^2  *C(7);
MddC1321 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)    *C(2);
MddC1322 = eigAvec1(1)  *eigAvec1(2)^2 *eigAvec1(3)    *C(5);
MddC1323 = eigAvec1(1)  *eigAvec1(2)  *eigAvec1(3)^2  *C(8);
MddC1331 = eigAvec1(1)^2            *eigAvec1(3)^2  *C(3);
MddC1332 = eigAvec1(1)  *eigAvec1(2)  *eigAvec1(3)^2  *C(6);
```

```
MddC1333 = eigAvec1(1)              *eigAvec1(3)^3   *C(9);

MddC13_e1 = (MddC1311 + MddC1312 + MddC1313 + MddC1321 + MddC1322 + MddC1323 + MddC1331 + MddC1332 +
MddC1333);

% Calculation of the 2,1 Component of the MddC 2nd Order Tensor

MddC21_e1 = MddC12_e1;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the MddC 2nd Order Tensor

MddC2211 = eigAvec1(1)^2 *eigAvec1(2)^2              *C(1);
MddC2212 = eigAvec1(1)   *eigAvec1(2)^3              *C(4);
MddC2213 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)    *C(7);
MddC2221 = eigAvec1(1)   *eigAvec1(2)^3              *C(2);
MddC2222 =               eigAvec1(2)^4              *C(5);
MddC2223 =               eigAvec1(2)^3 *eigAvec1(3)    *C(8);
MddC2231 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)    *C(3);
MddC2232 =               eigAvec1(2)^3 *eigAvec1(3)    *C(6);
MddC2233 =               eigAvec1(2)^2 *eigAvec1(3)^2   *C(9);

MddC22_e1 = (MddC2211 + MddC2212 + MddC2213 + MddC2221 + MddC2222 + MddC2223 + MddC2231 + MddC2232 +
MddC2233);

% Calculation of the 2,3 Component of the MddC 2nd Order Tensor

MddC2311 = eigAvec1(1)^2 *eigAvec1(2)   *eigAvec1(3)    *C(1);
MddC2312 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)    *C(4);
MddC2313 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *C(7);
MddC2321 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)    *C(2);
MddC2322 =               eigAvec1(2)^3 *eigAvec1(3)    *C(5);
MddC2323 =               eigAvec1(2)^2 *eigAvec1(3)^2   *C(8);
MddC2331 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *C(3);
MddC2332 =               eigAvec1(2)^2 *eigAvec1(3)^2   *C(6);
MddC2333 =               eigAvec1(2)   *eigAvec1(3)^3   *C(9);

MddC23_e1 = (MddC2311 + MddC2312 + MddC2313 + MddC2321 + MddC2322 + MddC2323 + MddC2331 + MddC2332 +
MddC2333);

% Calculation of the 3,1 Component of the MddC 2nd Order Tensor

MddC31_e1 = MddC13_e1;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the MddC 2nd Order Tensor

MddC32_e1 = MddC23_e1;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the MddC 2nd Order Tensor

MddC3311 = eigAvec1(1)^2              *eigAvec1(3)^2   *C(1);
MddC3312 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *C(4);
MddC3313 = eigAvec1(1)              *eigAvec1(3)^3   *C(7);
MddC3321 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *C(2);
MddC3322 =               eigAvec1(2)^2 *eigAvec1(3)^2   *C(5);
MddC3323 =               eigAvec1(2)   *eigAvec1(3)^3   *C(8);
MddC3331 = eigAvec1(1)              *eigAvec1(3)^3   *C(3);
MddC3332 =               eigAvec1(2)   *eigAvec1(3)^3   *C(6);
```

```
MddC3333 =                          eigAvec1(3)^4   *C(9);

MddC33_e1 = (MddC3311 + MddC3312 + MddC3313 + MddC3321 + MddC3322 + MddC3323 + MddC3331 + MddC3332 +
MddC3333);

%% CALCULATION FOR I = 2

% Calculation of the 1,1 Component of the MddC 2nd Order Tensor

MddC1111 = eigAvec2(1)^4                        *C(1);
MddC1112 = eigAvec2(1)^3 *eigAvec2(2)               *C(4);
MddC1113 = eigAvec2(1)^3            *eigAvec2(3)    *C(7);
MddC1121 = eigAvec2(1)^3 *eigAvec2(2)               *C(2);
MddC1122 = eigAvec2(1)^2 *eigAvec2(2)^2              *C(5);
MddC1123 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *C(8);
MddC1131 = eigAvec2(1)^3            *eigAvec2(3)    *C(3);
MddC1132 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *C(6);
MddC1133 = eigAvec2(1)^2            *eigAvec2(3)^2  *C(9);

MddC11_e2 = (MddC1111 + MddC1112 + MddC1113 + MddC1121 + MddC1122 + MddC1123 + MddC1131 + MddC1132 +
MddC1133);

% Calculation of the 1,2 Component of the MddC 2nd Order Tensor

MddC1211 = eigAvec2(1)^3 *eigAvec2(2)               *C(1);
MddC1212 = eigAvec2(1)^2 *eigAvec2(2)^2              *C(4);
MddC1213 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *C(7);
MddC1221 = eigAvec2(1)^2 *eigAvec2(2)^2              *C(2);
MddC1222 = eigAvec2(1)   *eigAvec2(2)^3             *C(5);
MddC1223 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *C(8);
MddC1231 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *C(3);
MddC1232 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *C(6);
MddC1233 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *C(9);

MddC12_e2 = (MddC1211 + MddC1212 + MddC1213 + MddC1221 + MddC1222 + MddC1223 + MddC1231 + MddC1232 +
MddC1233);

% Calculation of the 1,3 Component of the MddC 2nd Order Tensor

MddC1311 = eigAvec2(1)^3            *eigAvec2(3)    *C(1);
MddC1312 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *C(4);
MddC1313 = eigAvec2(1)^2            *eigAvec2(3)^2  *C(7);
MddC1321 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *C(2);
MddC1322 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *C(5);
MddC1323 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *C(8);
MddC1331 = eigAvec2(1)^2            *eigAvec2(3)^2  *C(3);
MddC1332 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *C(6);
MddC1333 = eigAvec2(1)              *eigAvec2(3)^3  *C(9);

MddC13_e2 = (MddC1311 + MddC1312 + MddC1313 + MddC1321 + MddC1322 + MddC1323 + MddC1331 + MddC1332 +
MddC1333);

% Calculation of the 2,1 Component of the MddC 2nd Order Tensor

MddC21_e2 = MddC12_e2;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the MddC 2nd Order Tensor
```

```
MddC2211 = eigAvec2(1)^2 *eigAvec2(2)^2                *C(1);
MddC2212 = eigAvec2(1)   *eigAvec2(2)^3                *C(4);
MddC2213 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *C(7);
MddC2221 = eigAvec2(1)   *eigAvec2(2)^3                *C(2);
MddC2222 =               eigAvec2(2)^4                 *C(5);
MddC2223 =               eigAvec2(2)^3 *eigAvec2(3)    *C(8);
MddC2231 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *C(3);
MddC2232 =               eigAvec2(2)^3 *eigAvec2(3)    *C(6);
MddC2233 =               eigAvec2(2)^2 *eigAvec2(3)^2   *C(9);

MddC22_e2 = (MddC2211 + MddC2212 + MddC2213 + MddC2221 + MddC2222 + MddC2223 + MddC2231 + MddC2232 +
MddC2233);

% Calculation of the 2,3 Component of the MddC 2nd Order Tensor

MddC2311 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *C(1);
MddC2312 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *C(4);
MddC2313 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *C(7);
MddC2321 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *C(2);
MddC2322 =               eigAvec2(2)^3 *eigAvec2(3)    *C(5);
MddC2323 =               eigAvec2(2)^2 *eigAvec2(3)^2  *C(8);
MddC2331 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *C(3);
MddC2332 =               eigAvec2(2)^2 *eigAvec2(3)^2  *C(6);
MddC2333 =               eigAvec2(2)   *eigAvec2(3)^3  *C(9);

MddC23_e2 = (MddC2311 + MddC2312 + MddC2313 + MddC2321 + MddC2322 + MddC2323 + MddC2331 + MddC2332 +
MddC2333);

% Calculation of the 3,1 Component of the MddC 2nd Order Tensor

MddC31_e2 = MddC13_e2;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the MddC 2nd Order Tensor

MddC32_e2 = MddC23_e2;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the MddC 2nd Order Tensor

MddC3311 = eigAvec2(1)^2               *eigAvec2(3)^2   *C(1);
MddC3312 = eigAvec2(1)   *eigAvec2(2)  *eigAvec2(3)^2   *C(4);
MddC3313 = eigAvec2(1)                 *eigAvec2(3)^3   *C(7);
MddC3321 = eigAvec2(1)   *eigAvec2(2)  *eigAvec2(3)^2   *C(2);
MddC3322 =               eigAvec2(2)^2 *eigAvec2(3)^2   *C(5);
MddC3323 =               eigAvec2(2)   *eigAvec2(3)^3   *C(8);
MddC3331 = eigAvec2(1)                 *eigAvec2(3)^3   *C(3);
MddC3332 =               eigAvec2(2)   *eigAvec2(3)^3   *C(6);
MddC3333 =                             eigAvec2(3)^4    *C(9);

MddC33_e2 = (MddC3311 + MddC3312 + MddC3313 + MddC3321 + MddC3322 + MddC3323 + MddC3331 + MddC3332 +
MddC3333);

%% CALCULATION FOR I = 3

% Calculation of the 1,1 Component of the MddC 2nd Order Tensor

MddC1111 = eigAvec3(1)^4                       *C(1);
```

```
MddC1112 = eigAvec3(1)^3 *eigAvec3(2)                *C(4);
MddC1113 = eigAvec3(1)^3             *eigAvec3(3)    *C(7);
MddC1121 = eigAvec3(1)^3 *eigAvec3(2)                *C(2);
MddC1122 = eigAvec3(1)^2 *eigAvec3(2)^2               *C(5);
MddC1123 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *C(8);
MddC1131 = eigAvec3(1)^3             *eigAvec3(3)    *C(3);
MddC1132 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *C(6);
MddC1133 = eigAvec3(1)^2             *eigAvec3(3)^2  *C(9);

MddC11_e3 = (MddC1111 + MddC1112 + MddC1113 + MddC1121 + MddC1122 + MddC1123 + MddC1131 + MddC1132 +
MddC1133);

% Calculation of the 1,2 Component of the MddC 2nd Order Tensor

MddC1211 = eigAvec3(1)^3 *eigAvec3(2)                *C(1);
MddC1212 = eigAvec3(1)^2 *eigAvec3(2)^2               *C(4);
MddC1213 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *C(7);
MddC1221 = eigAvec3(1)^2 *eigAvec3(2)^2               *C(2);
MddC1222 = eigAvec3(1)   *eigAvec3(2)^3               *C(5);
MddC1223 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *C(8);
MddC1231 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *C(3);
MddC1232 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *C(6);
MddC1233 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2  *C(9);

MddC12_e3 = (MddC1211 + MddC1212 + MddC1213 + MddC1221 + MddC1222 + MddC1223 + MddC1231 + MddC1232 +
MddC1233);

% Calculation of the 1,3 Component of the MddC 2nd Order Tensor

MddC1311 = eigAvec3(1)^3             *eigAvec3(3)    *C(1);
MddC1312 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *C(4);
MddC1313 = eigAvec3(1)^2             *eigAvec3(3)^2   *C(7);
MddC1321 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *C(2);
MddC1322 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *C(5);
MddC1323 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *C(8);
MddC1331 = eigAvec3(1)^2             *eigAvec3(3)^2   *C(3);
MddC1332 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *C(6);
MddC1333 = eigAvec3(1)               *eigAvec3(3)^3   *C(9);

MddC13_e3 = (MddC1311 + MddC1312 + MddC1313 + MddC1321 + MddC1322 + MddC1323 + MddC1331 + MddC1332 +
MddC1333);

% Calculation of the 2,1 Component of the MddC 2nd Order Tensor

MddC21_e3 = MddC12_e3;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the MddC 2nd Order Tensor

MddC2211 = eigAvec3(1)^2 *eigAvec3(2)^2               *C(1);
MddC2212 = eigAvec3(1)   *eigAvec3(2)^3               *C(4);
MddC2213 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *C(7);
MddC2221 = eigAvec3(1)   *eigAvec3(2)^3               *C(2);
MddC2222 =               eigAvec3(2)^4               *C(5);
MddC2223 =               eigAvec3(2)^3 *eigAvec3(3)    *C(8);
MddC2231 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *C(3);
MddC2232 =               eigAvec3(2)^3 *eigAvec3(3)    *C(6);
MddC2233 =               eigAvec3(2)^2 *eigAvec3(3)^2   *C(9);
```

```
MddC22_e3 = (MddC2211 + MddC2212 + MddC2213 + MddC2221 + MddC2222 + MddC2223 + MddC2231 + MddC2232 +
MddC2233);

% Calculation of the 2,3 Component of the MddC 2nd Order Tensor

MddC2311 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *C(1);
MddC2312 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *C(4);
MddC2313 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2  *C(7);
MddC2321 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *C(2);
MddC2322 =               eigAvec3(2)^3 *eigAvec3(3)     *C(5);
MddC2323 =               eigAvec3(2)^2 *eigAvec3(3)^2   *C(8);
MddC2331 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2  *C(3);
MddC2332 =               eigAvec3(2)^2 *eigAvec3(3)^2   *C(6);
MddC2333 =               eigAvec3(2)   *eigAvec3(3)^3   *C(9);

MddC23_e3 = (MddC2311 + MddC2312 + MddC2313 + MddC2321 + MddC2322 + MddC2323 + MddC2331 + MddC2332 +
MddC2333);

% Calculation of the 3,1 Component of the MddC 2nd Order Tensor

MddC31_e3 = MddC13_e3;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the MddC 2nd Order Tensor

MddC32_e3 = MddC23_e3;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the MddC 2nd Order Tensor

MddC3311 = eigAvec3(1)^2              *eigAvec3(3)^2   *C(1);
MddC3312 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *C(4);
MddC3313 = eigAvec3(1)                *eigAvec3(3)^3   *C(7);
MddC3321 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *C(2);
MddC3322 =               eigAvec3(2)^2 *eigAvec3(3)^2   *C(5);
MddC3323 =               eigAvec3(2)   *eigAvec3(3)^3   *C(8);
MddC3331 = eigAvec3(1)                *eigAvec3(3)^3   *C(3);
MddC3332 =               eigAvec3(2)   *eigAvec3(3)^3   *C(6);
MddC3333 =                             eigAvec3(3)^4   *C(9);

MddC33_e3 = (MddC3311 + MddC3312 + MddC3313 + MddC3321 + MddC3322 + MddC3323 + MddC3331 + MddC3332 +
MddC3333);

%% Summing the terms over i (i.e. i = 1,2,3)

MddC11 = MddC11_e1 + MddC11_e2 + MddC11_e3;
MddC12 = MddC12_e1 + MddC12_e2 + MddC12_e3;
MddC13 = MddC13_e1 + MddC13_e2 + MddC13_e3;
MddC21 = MddC21_e1 + MddC21_e2 + MddC21_e3;
MddC22 = MddC22_e1 + MddC22_e2 + MddC22_e3;
MddC23 = MddC23_e1 + MddC23_e2 + MddC23_e3;
MddC31 = MddC32_e1 + MddC31_e2 + MddC31_e3;
MddC32 = MddC32_e1 + MddC32_e2 + MddC32_e3;
MddC33 = MddC33_e1 + MddC33_e2 + MddC33_e3;

%% Forming the M:C Tensor (M:C)

MddC = [MddC11 MddC12 MddC13 MddC21 MddC22 MddC23 MddC31 MddC32 MddC33];
```

```
end
```

E.3.7 MddAddC.m

```
function [MddAddC] = MddAddC(C,A)

% Function written to compute the 4th order M tensor double dotted into the
% 4th order A tensor double dotted into the 2nd order anistropy tensor (D).

% 4th Order A tensor used the quadratic closure (AA)

[eigAvec,eigAval] = eig(reshape(A,3,3));   % Calculation of the numerical eigenvalues and eigenvectors.

eigAvec = fliplr(eigAvec);
eigAval = rot90(rot90(eigAval));

% Assigning the numerical values to discrete variables to ease in
% calculation.

eigAvec1 = eigAvec(:,1);
eigAvec2 = eigAvec(:,2);
eigAvec3 = eigAvec(:,3);

%% CALCULATION FOR I = 1

% Calculation of the 1,1 Component of the LddD 2nd Order Tensor

MddAddC1111 = eigAvec1(1)^4                          *A(1);
MddAddC1112 = eigAvec1(1)^3 *eigAvec1(2)             *A(4);
MddAddC1113 = eigAvec1(1)^3            *eigAvec1(3)  *A(7);
MddAddC1121 = eigAvec1(1)^3 *eigAvec1(2)             *A(2);
MddAddC1122 = eigAvec1(1)^2 *eigAvec1(2)^2           *A(5);
MddAddC1123 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)  *A(8);
MddAddC1131 = eigAvec1(1)^3            *eigAvec1(3)  *A(3);
MddAddC1132 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)  *A(6);
MddAddC1133 = eigAvec1(1)^2            *eigAvec1(3)^2  *A(9);

MddAddC11_e1 = Explicit_Quad_v(C,A)*(MddAddC1111 + MddAddC1112 + MddAddC1113 + MddAddC1121 +
MddAddC1122 + MddAddC1123 + MddAddC1131 + MddAddC1132 + MddAddC1133);

% Calculation of the 1,2 Component of the MddAddC 2nd Order Tensor

MddAddC1211 = eigAvec1(1)^3 *eigAvec1(2)             *A(1);
MddAddC1212 = eigAvec1(1)^2 *eigAvec1(2)^2           *A(4);
MddAddC1213 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)  *A(7);
MddAddC1221 = eigAvec1(1)^2 *eigAvec1(2)^2           *A(2);
MddAddC1222 = eigAvec1(1)   *eigAvec1(2)^3           *A(5);
MddAddC1223 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)  *A(8);
MddAddC1231 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)  *A(3);
MddAddC1232 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)  *A(6);
MddAddC1233 = eigAvec1(1)   *eigAvec1(2)  *eigAvec1(3)^2  *A(9);

MddAddC12_e1 = Explicit_Quad_v(C,A)*(MddAddC1211 + MddAddC1212 + MddAddC1213 + MddAddC1221 +
MddAddC1222 + MddAddC1223 + MddAddC1231 + MddAddC1232 + MddAddC1233);

% Calculation of the 1,3 Component of the MddAddC 2nd Order Tensor
```

```
MddAddC1311 = eigAvec1(1)^3              *eigAvec1(3)    *A(1);
MddAddC1312 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)    *A(4);
MddAddC1313 = eigAvec1(1)^2              *eigAvec1(3)^2  *A(7);
MddAddC1321 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)    *A(2);
MddAddC1322 = eigAvec1(1)  *eigAvec1(2)^2 *eigAvec1(3)    *A(5);
MddAddC1323 = eigAvec1(1)  *eigAvec1(2)  *eigAvec1(3)^2  *A(8);
MddAddC1331 = eigAvec1(1)^2              *eigAvec1(3)^2  *A(3);
MddAddC1332 = eigAvec1(1)  *eigAvec1(2)  *eigAvec1(3)^2  *A(6);
MddAddC1333 = eigAvec1(1)               *eigAvec1(3)^3  *A(9);

MddAddC13_e1 = Explicit_Quad_v(C,A)*(MddAddC1311 + MddAddC1312 + MddAddC1313 + MddAddC1321 +
MddAddC1322 + MddAddC1323 + MddAddC1331 + MddAddC1332 + MddAddC1333);

% Calculation of the 2,1 Component of the MddAddC 2nd Order Tensor

MddAddC21_e1 = MddAddC12_e1;   %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the MddAddC 2nd Order Tensor

MddAddC2211 = eigAvec1(1)^2 *eigAvec1(2)^2               *A(1);
MddAddC2212 = eigAvec1(1)  *eigAvec1(2)^3               *A(4);
MddAddC2213 = eigAvec1(1)  *eigAvec1(2)^2 *eigAvec1(3)    *A(7);
MddAddC2221 = eigAvec1(1)  *eigAvec1(2)^3               *A(2);
MddAddC2222 =             eigAvec1(2)^4               *A(5);
MddAddC2223 =             eigAvec1(2)^3 *eigAvec1(3)    *A(8);
MddAddC2231 = eigAvec1(1)  *eigAvec1(2)^2 *eigAvec1(3)    *A(3);
MddAddC2232 =             eigAvec1(2)^3 *eigAvec1(3)    *A(6);
MddAddC2233 =             eigAvec1(2)^2 *eigAvec1(3)^2  *A(9);

MddAddC22_e1 = Explicit_Quad_v(C,A)*(MddAddC2211 + MddAddC2212 + MddAddC2213 + MddAddC2221 +
MddAddC2222 + MddAddC2223 + MddAddC2231 + MddAddC2232 + MddAddC2233);

% Calculation of the 2,3 Component of the MddAddC 2nd Order Tensor

MddAddC2311 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)    *A(1);
MddAddC2312 = eigAvec1(1)  *eigAvec1(2)^2 *eigAvec1(3)    *A(4);
MddAddC2313 = eigAvec1(1)  *eigAvec1(2)  *eigAvec1(3)^2  *A(7);
MddAddC2321 = eigAvec1(1)  *eigAvec1(2)^2 *eigAvec1(3)    *A(2);
MddAddC2322 =             eigAvec1(2)^3 *eigAvec1(3)    *A(5);
MddAddC2323 =             eigAvec1(2)^2 *eigAvec1(3)^2  *A(8);
MddAddC2331 = eigAvec1(1)  *eigAvec1(2)  *eigAvec1(3)^2  *A(3);
MddAddC2332 =             eigAvec1(2)^2 *eigAvec1(3)^2  *A(6);
MddAddC2333 =             eigAvec1(2)  *eigAvec1(3)^3  *A(9);

MddAddC23_e1 = Explicit_Quad_v(C,A)*(MddAddC2311 + MddAddC2312 + MddAddC2313 + MddAddC2321 +
MddAddC2322 + MddAddC2323 + MddAddC2331 + MddAddC2332 + MddAddC2333);

% Calculation of the 3,1 Component of the MddAddC 2nd Order Tensor

MddAddC31_e1 = MddAddC13_e1;   %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the MddAddC 2nd Order Tensor

MddAddC32_e1 = MddAddC23_e1;   %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the MddAddC 2nd Order Tensor
```

```
MddAddC3311 = eigAvec1(1)^2              *eigAvec1(3)^2   *A(1);
MddAddC3312 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *A(4);
MddAddC3313 = eigAvec1(1)              *eigAvec1(3)^3   *A(7);
MddAddC3321 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *A(2);
MddAddC3322 =              eigAvec1(2)^2 *eigAvec1(3)^2   *A(5);
MddAddC3323 =              eigAvec1(2)   *eigAvec1(3)^3   *A(8);
MddAddC3331 = eigAvec1(1)              *eigAvec1(3)^3   *A(3);
MddAddC3332 =              eigAvec1(2)   *eigAvec1(3)^3   *A(6);
MddAddC3333 =                        eigAvec1(3)^4   *A(9);

MddAddC33_e1 = Explicit_Quad_v(C,A)*(MddAddC3311 + MddAddC3312 + MddAddC3313 + MddAddC3321 +
MddAddC3322 + MddAddC3323 + MddAddC3331 + MddAddC3332 + MddAddC3333);

%% CALCULATION FOR I = 2

% Calculation of the 1,1 Component of the LddD 2nd Order Tensor

MddAddC1111 = eigAvec2(1)^4                        *A(1);
MddAddC1112 = eigAvec2(1)^3 *eigAvec2(2)              *A(4);
MddAddC1113 = eigAvec2(1)^3            *eigAvec2(3)    *A(7);
MddAddC1121 = eigAvec2(1)^3 *eigAvec2(2)              *A(2);
MddAddC1122 = eigAvec2(1)^2 *eigAvec2(2)^2             *A(5);
MddAddC1123 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *A(8);
MddAddC1131 = eigAvec2(1)^3            *eigAvec2(3)    *A(3);
MddAddC1132 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *A(6);
MddAddC1133 = eigAvec2(1)^2            *eigAvec2(3)^2  *A(9);

MddAddC11_e2 = Explicit_Quad_v(C,A)*(MddAddC1111 + MddAddC1112 + MddAddC1113 + MddAddC1121 +
MddAddC1122 + MddAddC1123 + MddAddC1131 + MddAddC1132 + MddAddC1133);

% Calculation of the 1,2 Component of the MddAddC 2nd Order Tensor

MddAddC1211 = eigAvec2(1)^3 *eigAvec2(2)              *A(1);
MddAddC1212 = eigAvec2(1)^2 *eigAvec2(2)^2             *A(4);
MddAddC1213 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *A(7);
MddAddC1221 = eigAvec2(1)^2 *eigAvec2(2)^2             *A(2);
MddAddC1222 = eigAvec2(1)   *eigAvec2(2)^3             *A(5);
MddAddC1223 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *A(8);
MddAddC1231 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *A(3);
MddAddC1232 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *A(6);
MddAddC1233 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *A(9);

MddAddC12_e2 = Explicit_Quad_v(C,A)*(MddAddC1211 + MddAddC1212 + MddAddC1213 + MddAddC1221 +
MddAddC1222 + MddAddC1223 + MddAddC1231 + MddAddC1232 + MddAddC1233);

% Calculation of the 1,3 Component of the MddAddC 2nd Order Tensor

MddAddC1311 = eigAvec2(1)^3            *eigAvec2(3)    *A(1);
MddAddC1312 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *A(4);
MddAddC1313 = eigAvec2(1)^2            *eigAvec2(3)^2   *A(7);
MddAddC1321 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *A(2);
MddAddC1322 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *A(5);
MddAddC1323 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *A(8);
MddAddC1331 = eigAvec2(1)^2            *eigAvec2(3)^2   *A(3);
MddAddC1332 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *A(6);
MddAddC1333 = eigAvec2(1)              *eigAvec2(3)^3   *A(9);
```

```
MddAddC13_e2 = Explicit_Quad_v(C,A)*(MddAddC1311 + MddAddC1312 + MddAddC1313 + MddAddC1321 +
MddAddC1322 + MddAddC1323 + MddAddC1331 + MddAddC1332 + MddAddC1333);

% Calculation of the 2,1 Component of the MddAddC 2nd Order Tensor

MddAddC21_e2 = MddAddC12_e2;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the MddAddC 2nd Order Tensor

MddAddC2211 = eigAvec2(1)^2 *eigAvec2(2)^2                *A(1);
MddAddC2212 = eigAvec2(1)   *eigAvec2(2)^3                *A(4);
MddAddC2213 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)   *A(7);
MddAddC2221 = eigAvec2(1)   *eigAvec2(2)^3                *A(2);
MddAddC2222 =               eigAvec2(2)^4                 *A(5);
MddAddC2223 =               eigAvec2(2)^3 *eigAvec2(3)    *A(8);
MddAddC2231 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)   *A(3);
MddAddC2232 =               eigAvec2(2)^3 *eigAvec2(3)    *A(6);
MddAddC2233 =               eigAvec2(2)^2 *eigAvec2(3)^2  *A(9);

MddAddC22_e2 = Explicit_Quad_v(C,A)*(MddAddC2211 + MddAddC2212 + MddAddC2213 + MddAddC2221 +
MddAddC2222 + MddAddC2223 + MddAddC2231 + MddAddC2232 + MddAddC2233);

% Calculation of the 2,3 Component of the MddAddC 2nd Order Tensor

MddAddC2311 = eigAvec2(1)^2 *eigAvec2(2)  *eigAvec2(3)    *A(1);
MddAddC2312 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)   *A(4);
MddAddC2313 = eigAvec2(1)   *eigAvec2(2)  *eigAvec2(3)^2  *A(7);
MddAddC2321 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)   *A(2);
MddAddC2322 =               eigAvec2(2)^3 *eigAvec2(3)    *A(5);
MddAddC2323 =               eigAvec2(2)^2 *eigAvec2(3)^2  *A(8);
MddAddC2331 = eigAvec2(1)   *eigAvec2(2)  *eigAvec2(3)^2  *A(3);
MddAddC2332 =               eigAvec2(2)^2 *eigAvec2(3)^2  *A(6);
MddAddC2333 =               eigAvec2(2)   *eigAvec2(3)^3  *A(9);

MddAddC23_e2 = Explicit_Quad_v(C,A)*(MddAddC2311 + MddAddC2312 + MddAddC2313 + MddAddC2321 +
MddAddC2322 + MddAddC2323 + MddAddC2331 + MddAddC2332 + MddAddC2333);

% Calculation of the 3,1 Component of the MddAddC 2nd Order Tensor

MddAddC31_e2 = MddAddC13_e2;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the MddAddC 2nd Order Tensor

MddAddC32_e2 = MddAddC23_e2;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the MddAddC 2nd Order Tensor

MddAddC3311 = eigAvec2(1)^2               *eigAvec2(3)^2  *A(1);
MddAddC3312 = eigAvec2(1)   *eigAvec2(2)  *eigAvec2(3)^2  *A(4);
MddAddC3313 = eigAvec2(1)                 *eigAvec2(3)^3  *A(7);
MddAddC3321 = eigAvec2(1)   *eigAvec2(2)  *eigAvec2(3)^2  *A(2);
MddAddC3322 =               eigAvec2(2)^2 *eigAvec2(3)^2  *A(5);
MddAddC3323 =               eigAvec2(2)   *eigAvec2(3)^3  *A(8);
MddAddC3331 = eigAvec2(1)                 *eigAvec2(3)^3  *A(3);
MddAddC3332 =               eigAvec2(2)   *eigAvec2(3)^3  *A(6);
MddAddC3333 =                             eigAvec2(3)^4   *A(9);
```

```
MddAddC33_e2 = Explicit_Quad_v(C,A)*(MddAddC3311 + MddAddC3312 + MddAddC3313 + MddAddC3321 +
MddAddC3322 + MddAddC3323 + MddAddC3331 + MddAddC3332 + MddAddC3333);

%% CALCULATION FOR I = 3

% Calculation of the 1,1 Component of the LddD 2nd Order Tensor

MddAddC1111 = eigAvec3(1)^4                         *A(1);
MddAddC1112 = eigAvec3(1)^3 *eigAvec3(2)            *A(4);
MddAddC1113 = eigAvec3(1)^3          *eigAvec3(3)   *A(7);
MddAddC1121 = eigAvec3(1)^3 *eigAvec3(2)            *A(2);
MddAddC1122 = eigAvec3(1)^2 *eigAvec3(2)^2          *A(5);
MddAddC1123 = eigAvec3(1)^2 *eigAvec3(2)  *eigAvec3(3)   *A(8);
MddAddC1131 = eigAvec3(1)^3          *eigAvec3(3)   *A(3);
MddAddC1132 = eigAvec3(1)^2 *eigAvec3(2)  *eigAvec3(3)   *A(6);
MddAddC1133 = eigAvec3(1)^2          *eigAvec3(3)^2 *A(9);

MddAddC11_e3 = Explicit_Quad_v(C,A)*(MddAddC1111 + MddAddC1112 + MddAddC1113 + MddAddC1121 +
MddAddC1122 + MddAddC1123 + MddAddC1131 + MddAddC1132 + MddAddC1133);

% Calculation of the 1,2 Component of the MddAddC 2nd Order Tensor

MddAddC1211 = eigAvec3(1)^3 *eigAvec3(2)            *A(1);
MddAddC1212 = eigAvec3(1)^2 *eigAvec3(2)^2          *A(4);
MddAddC1213 = eigAvec3(1)^2 *eigAvec3(2)  *eigAvec3(3)   *A(7);
MddAddC1221 = eigAvec3(1)^2 *eigAvec3(2)^2          *A(2);
MddAddC1222 = eigAvec3(1)   *eigAvec3(2)^3          *A(5);
MddAddC1223 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)   *A(8);
MddAddC1231 = eigAvec3(1)^2 *eigAvec3(2)  *eigAvec3(3)   *A(3);
MddAddC1232 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)   *A(6);
MddAddC1233 = eigAvec3(1)   *eigAvec3(2)  *eigAvec3(3)^2 *A(9);

MddAddC12_e3 = Explicit_Quad_v(C,A)*(MddAddC1211 + MddAddC1212 + MddAddC1213 + MddAddC1221 +
MddAddC1222 + MddAddC1223 + MddAddC1231 + MddAddC1232 + MddAddC1233);

% Calculation of the 1,3 Component of the MddAddC 2nd Order Tensor

MddAddC1311 = eigAvec3(1)^3          *eigAvec3(3)   *A(1);
MddAddC1312 = eigAvec3(1)^2 *eigAvec3(2)  *eigAvec3(3)   *A(4);
MddAddC1313 = eigAvec3(1)^2          *eigAvec3(3)^2 *A(7);
MddAddC1321 = eigAvec3(1)^2 *eigAvec3(2)  *eigAvec3(3)   *A(2);
MddAddC1322 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)   *A(5);
MddAddC1323 = eigAvec3(1)   *eigAvec3(2)  *eigAvec3(3)^2 *A(8);
MddAddC1331 = eigAvec3(1)^2          *eigAvec3(3)^2 *A(3);
MddAddC1332 = eigAvec3(1)   *eigAvec3(2)  *eigAvec3(3)^2 *A(6);
MddAddC1333 = eigAvec3(1)            *eigAvec3(3)^3 *A(9);

MddAddC13_e3 = Explicit_Quad_v(C,A)*(MddAddC1311 + MddAddC1312 + MddAddC1313 + MddAddC1321 +
MddAddC1322 + MddAddC1323 + MddAddC1331 + MddAddC1332 + MddAddC1333);

% Calculation of the 2,1 Component of the MddAddC 2nd Order Tensor

MddAddC21_e3 = MddAddC12_e3;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the MddAddC 2nd Order Tensor

MddAddC2211 = eigAvec3(1)^2 *eigAvec3(2)^2          *A(1);
```

```
MddAddC2212 = eigAvec3(1)   *eigAvec3(2)^3               *A(4);
MddAddC2213 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *A(7);
MddAddC2221 = eigAvec3(1)   *eigAvec3(2)^3               *A(2);
MddAddC2222 =               eigAvec3(2)^4              *A(5);
MddAddC2223 =               eigAvec3(2)^3 *eigAvec3(3)    *A(8);
MddAddC2231 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *A(3);
MddAddC2232 =               eigAvec3(2)^3 *eigAvec3(3)    *A(6);
MddAddC2233 =               eigAvec3(2)^2 *eigAvec3(3)^2   *A(9);

MddAddC22_e3 = Explicit_Quad_v(C,A)*(MddAddC2211 + MddAddC2212 + MddAddC2213 + MddAddC2221 +
MddAddC2222 + MddAddC2223 + MddAddC2231 + MddAddC2232 + MddAddC2233);

% Calculation of the 2,3 Component of the MddAddC 2nd Order Tensor

MddAddC2311 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *A(1);
MddAddC2312 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *A(4);
MddAddC2313 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *A(7);
MddAddC2321 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *A(2);
MddAddC2322 =               eigAvec3(2)^3 *eigAvec3(3)    *A(5);
MddAddC2323 =               eigAvec3(2)^2 *eigAvec3(3)^2   *A(8);
MddAddC2331 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *A(3);
MddAddC2332 =               eigAvec3(2)^2 *eigAvec3(3)^2   *A(6);
MddAddC2333 =               eigAvec3(2)   *eigAvec3(3)^3   *A(9);

MddAddC23_e3 = Explicit_Quad_v(C,A)*(MddAddC2311 + MddAddC2312 + MddAddC2313 + MddAddC2321 +
MddAddC2322 + MddAddC2323 + MddAddC2331 + MddAddC2332 + MddAddC2333);

% Calculation of the 3,1 Component of the MddAddC 2nd Order Tensor

MddAddC31_e3 = MddAddC13_e3;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the MddAddC 2nd Order Tensor

MddAddC32_e3 = MddAddC23_e3;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the MddAddC 2nd Order Tensor

MddAddC3311 = eigAvec3(1)^2               *eigAvec3(3)^2   *A(1);
MddAddC3312 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *A(4);
MddAddC3313 = eigAvec3(1)                *eigAvec3(3)^3   *A(7);
MddAddC3321 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *A(2);
MddAddC3322 =               eigAvec3(2)^2 *eigAvec3(3)^2   *A(5);
MddAddC3323 =               eigAvec3(2)   *eigAvec3(3)^3   *A(8);
MddAddC3331 = eigAvec3(1)                *eigAvec3(3)^3   *A(3);
MddAddC3332 =               eigAvec3(2)   *eigAvec3(3)^3   *A(6);
MddAddC3333 =                           eigAvec3(3)^4   *A(9);

MddAddC33_e3 = Explicit_Quad_v(C,A)*(MddAddC3311 + MddAddC3312 + MddAddC3313 + MddAddC3321 +
MddAddC3322 + MddAddC3323 + MddAddC3331 + MddAddC3332 + MddAddC3333);

%% Summing the terms over i (i.e. i = 1,2,3)

MddAddC11 = MddAddC11_e1 + MddAddC11_e2 + MddAddC11_e3;
MddAddC12 = MddAddC12_e1 + MddAddC12_e2 + MddAddC12_e3;
MddAddC13 = MddAddC13_e1 + MddAddC13_e2 + MddAddC13_e3;
MddAddC21 = MddAddC21_e1 + MddAddC21_e2 + MddAddC21_e3;
MddAddC22 = MddAddC22_e1 + MddAddC22_e2 + MddAddC22_e3;
```

```
MddAddC23 = MddAddC23_e1 + MddAddC23_e2 + MddAddC23_e3;
MddAddC31 = MddAddC32_e1 + MddAddC31_e2 + MddAddC31_e3;
MddAddC32 = MddAddC32_e1 + MddAddC32_e2 + MddAddC32_e3;
MddAddC33 = MddAddC33_e1 + MddAddC33_e2 + MddAddC33_e3;

%% Forming the M:A:C Tensor (2nd Order)

MddAddC = [MddAddC11 MddAddC12 MddAddC13 MddAddC21 MddAddC22 MddAddC23 MddAddC31 MddAddC32
MddAddC33];
```

E.3.8 MddAddC_IBOF.m

```
function [MddAddC] = MddAddC_IBOF(C,A)

% Function written to compute the 4th order M tensor double dotted into the
% 4th Order A tensor double dotted into 2nd order anisotropy tensor (D).

% 4th Order A Tensor Used IBOF Closure Approximation.

[eigAvec,eigAval] = eig(reshape(A,3,3));   % Calculation of the numerical eigenvalues and eigenvectors.

eigAvec = fliplr(eigAvec);
eigAval = rot90(rot90(eigAval));

% Assigning the numerical values to discrete variables to ease in
% calculation.

eigAvec1 = eigAvec(:,1);
eigAvec2 = eigAvec(:,2);
eigAvec3 = eigAvec(:,3);

%% CALCULATION FOR I = 1

% Calculation of the 1,1 Component of the LddD 2nd Order Tensor

MddAddC1111 = eigAvec1(1)^4                          *Explicit_IBOF_v(C,A,1,1);
MddAddC1112 = eigAvec1(1)^3 *eigAvec1(2)             *Explicit_IBOF_v(C,A,2,1);
MddAddC1113 = eigAvec1(1)^3            *eigAvec1(3)  *Explicit_IBOF_v(C,A,3,1);
MddAddC1121 = eigAvec1(1)^3 *eigAvec1(2)             *Explicit_IBOF_v(C,A,1,2);
MddAddC1122 = eigAvec1(1)^2 *eigAvec1(2)^2           *Explicit_IBOF_v(C,A,2,2);
MddAddC1123 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)  *Explicit_IBOF_v(C,A,3,2);
MddAddC1131 = eigAvec1(1)^3            *eigAvec1(3)  *Explicit_IBOF_v(C,A,1,3);
MddAddC1132 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)  *Explicit_IBOF_v(C,A,2,3);
MddAddC1133 = eigAvec1(1)^2            *eigAvec1(3)^2 *Explicit_IBOF_v(C,A,3,3);

MddAddC11_e1 = (MddAddC1111 + MddAddC1112 + MddAddC1113 + MddAddC1121 + MddAddC1122 + MddAddC1123 +
MddAddC1131 + MddAddC1132 + MddAddC1133);

% Calculation of the 1,2 Component of the MddAddC 2nd Order Tensor

MddAddC1211 = eigAvec1(1)^3 *eigAvec1(2)             *Explicit_IBOF_v(C,A,1,1);
MddAddC1212 = eigAvec1(1)^2 *eigAvec1(2)^2           *Explicit_IBOF_v(C,A,2,1);
MddAddC1213 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)  *Explicit_IBOF_v(C,A,3,1);
MddAddC1221 = eigAvec1(1)^2 *eigAvec1(2)^2           *Explicit_IBOF_v(C,A,1,2);
MddAddC1222 = eigAvec1(1)  *eigAvec1(2)^3            *Explicit_IBOF_v(C,A,2,2);
MddAddC1223 = eigAvec1(1)  *eigAvec1(2)^2 *eigAvec1(3)  *Explicit_IBOF_v(C,A,3,2);
MddAddC1231 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)  *Explicit_IBOF_v(C,A,1,3);
MddAddC1232 = eigAvec1(1)  *eigAvec1(2)^2 *eigAvec1(3)  *Explicit_IBOF_v(C,A,2,3);
```

```
MddAddC1233 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2  *Explicit_IBOF_v(C,A,3,3);

MddAddC12_e1 = (MddAddC1211 + MddAddC1212 + MddAddC1213 + MddAddC1221 + MddAddC1222 + MddAddC1223 +
MddAddC1231 + MddAddC1232 + MddAddC1233);

% Calculation of the 1,3 Component of the MddAddC 2nd Order Tensor

MddAddC1311 = eigAvec1(1)^3              *eigAvec1(3)     *Explicit_IBOF_v(C,A,1,1);
MddAddC1312 = eigAvec1(1)^2 *eigAvec1(2)   *eigAvec1(3)     *Explicit_IBOF_v(C,A,2,1);
MddAddC1313 = eigAvec1(1)^2              *eigAvec1(3)^2   *Explicit_IBOF_v(C,A,3,1);
MddAddC1321 = eigAvec1(1)^2 *eigAvec1(2)   *eigAvec1(3)     *Explicit_IBOF_v(C,A,1,2);
MddAddC1322 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)     *Explicit_IBOF_v(C,A,2,2);
MddAddC1323 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *Explicit_IBOF_v(C,A,3,2);
MddAddC1331 = eigAvec1(1)^2              *eigAvec1(3)^2   *Explicit_IBOF_v(C,A,1,3);
MddAddC1332 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *Explicit_IBOF_v(C,A,2,3);
MddAddC1333 = eigAvec1(1)                *eigAvec1(3)^3   *Explicit_IBOF_v(C,A,3,3);

MddAddC13_e1 = (MddAddC1311 + MddAddC1312 + MddAddC1313 + MddAddC1321 + MddAddC1322 + MddAddC1323 +
MddAddC1331 + MddAddC1332 + MddAddC1333);

% Calculation of the 2,1 Component of the MddAddC 2nd Order Tensor

MddAddC21_e1 = MddAddC12_e1;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the MddAddC 2nd Order Tensor

MddAddC2211 = eigAvec1(1)^2 *eigAvec1(2)^2               *Explicit_IBOF_v(C,A,1,1);
MddAddC2212 = eigAvec1(1)   *eigAvec1(2)^3               *Explicit_IBOF_v(C,A,2,1);
MddAddC2213 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)     *Explicit_IBOF_v(C,A,3,1);
MddAddC2221 = eigAvec1(1)   *eigAvec1(2)^3               *Explicit_IBOF_v(C,A,1,2);
MddAddC2222 =               eigAvec1(2)^4                *Explicit_IBOF_v(C,A,2,2);
MddAddC2223 =               eigAvec1(2)^3 *eigAvec1(3)     *Explicit_IBOF_v(C,A,3,2);
MddAddC2231 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)     *Explicit_IBOF_v(C,A,1,3);
MddAddC2232 =               eigAvec1(2)^3 *eigAvec1(3)     *Explicit_IBOF_v(C,A,2,3);
MddAddC2233 =               eigAvec1(2)^2 *eigAvec1(3)^2   *Explicit_IBOF_v(C,A,3,3);

MddAddC22_e1 = (MddAddC2211 + MddAddC2212 + MddAddC2213 + MddAddC2221 + MddAddC2222 + MddAddC2223 +
MddAddC2231 + MddAddC2232 + MddAddC2233);

% Calculation of the 2,3 Component of the MddAddC 2nd Order Tensor

MddAddC2311 = eigAvec1(1)^2 *eigAvec1(2)   *eigAvec1(3)     *Explicit_IBOF_v(C,A,1,1);
MddAddC2312 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)     *Explicit_IBOF_v(C,A,2,1);
MddAddC2313 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *Explicit_IBOF_v(C,A,3,1);
MddAddC2321 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)     *Explicit_IBOF_v(C,A,1,2);
MddAddC2322 =               eigAvec1(2)^3 *eigAvec1(3)     *Explicit_IBOF_v(C,A,2,2);
MddAddC2323 =               eigAvec1(2)^2 *eigAvec1(3)^2   *Explicit_IBOF_v(C,A,3,2);
MddAddC2331 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *Explicit_IBOF_v(C,A,1,3);
MddAddC2332 =               eigAvec1(2)^2 *eigAvec1(3)^2   *Explicit_IBOF_v(C,A,2,3);
MddAddC2333 =               eigAvec1(2)   *eigAvec1(3)^3   *Explicit_IBOF_v(C,A,3,3);

MddAddC23_e1 = (MddAddC2311 + MddAddC2312 + MddAddC2313 + MddAddC2321 + MddAddC2322 + MddAddC2323 +
MddAddC2331 + MddAddC2332 + MddAddC2333);

% Calculation of the 3,1 Component of the MddAddC 2nd Order Tensor

MddAddC31_e1 = MddAddC13_e1;    %DUE TO SYMMETRY!
```

```
% Calculation of the 3,2 Component of the MddAddC 2nd Order Tensor

MddAddC32_e1 = MddAddC23_e1;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the MddAddC 2nd Order Tensor

MddAddC3311 = eigAvec1(1)^2              *eigAvec1(3)^2   *Explicit_IBOF_v(C,A,1,1);
MddAddC3312 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *Explicit_IBOF_v(C,A,2,1);
MddAddC3313 = eigAvec1(1)              *eigAvec1(3)^3   *Explicit_IBOF_v(C,A,3,1);
MddAddC3321 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *Explicit_IBOF_v(C,A,1,2);
MddAddC3322 =             eigAvec1(2)^2 *eigAvec1(3)^2   *Explicit_IBOF_v(C,A,2,2);
MddAddC3323 =             eigAvec1(2)   *eigAvec1(3)^3   *Explicit_IBOF_v(C,A,3,2);
MddAddC3331 = eigAvec1(1)              *eigAvec1(3)^3   *Explicit_IBOF_v(C,A,1,3);
MddAddC3332 =             eigAvec1(2)   *eigAvec1(3)^3   *Explicit_IBOF_v(C,A,2,3);
MddAddC3333 =                        eigAvec1(3)^4   *Explicit_IBOF_v(C,A,3,3);

MddAddC33_e1 = (MddAddC3311 + MddAddC3312 + MddAddC3313 + MddAddC3321 + MddAddC3322 + MddAddC3323 +
MddAddC3331 + MddAddC3332 + MddAddC3333);

%% CALCULATION FOR I = 2

% Calculation of the 1,1 Component of the LddD 2nd Order Tensor

MddAddC1111 = eigAvec2(1)^4                        *Explicit_IBOF_v(C,A,1,1);
MddAddC1112 = eigAvec2(1)^3 *eigAvec2(2)              *Explicit_IBOF_v(C,A,2,1);
MddAddC1113 = eigAvec2(1)^3             *eigAvec2(3)    *Explicit_IBOF_v(C,A,3,1);
MddAddC1121 = eigAvec2(1)^3 *eigAvec2(2)              *Explicit_IBOF_v(C,A,1,2);
MddAddC1122 = eigAvec2(1)^2 *eigAvec2(2)^2             *Explicit_IBOF_v(C,A,2,2);
MddAddC1123 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *Explicit_IBOF_v(C,A,3,2);
MddAddC1131 = eigAvec2(1)^3             *eigAvec2(3)    *Explicit_IBOF_v(C,A,1,3);
MddAddC1132 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *Explicit_IBOF_v(C,A,2,3);
MddAddC1133 = eigAvec2(1)^2             *eigAvec2(3)^2  *Explicit_IBOF_v(C,A,3,3);

MddAddC11_e2 = (MddAddC1111 + MddAddC1112 + MddAddC1113 + MddAddC1121 + MddAddC1122 + MddAddC1123 +
MddAddC1131 + MddAddC1132 + MddAddC1133);

% Calculation of the 1,2 Component of the MddAddC 2nd Order Tensor

MddAddC1211 = eigAvec2(1)^3 *eigAvec2(2)              *Explicit_IBOF_v(C,A,1,1);
MddAddC1212 = eigAvec2(1)^2 *eigAvec2(2)^2             *Explicit_IBOF_v(C,A,2,1);
MddAddC1213 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *Explicit_IBOF_v(C,A,3,1);
MddAddC1221 = eigAvec2(1)^2 *eigAvec2(2)^2             *Explicit_IBOF_v(C,A,1,2);
MddAddC1222 = eigAvec2(1)   *eigAvec2(2)^3             *Explicit_IBOF_v(C,A,2,2);
MddAddC1223 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *Explicit_IBOF_v(C,A,3,2);
MddAddC1231 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *Explicit_IBOF_v(C,A,1,3);
MddAddC1232 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *Explicit_IBOF_v(C,A,2,3);
MddAddC1233 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *Explicit_IBOF_v(C,A,3,3);

MddAddC12_e2 = (MddAddC1211 + MddAddC1212 + MddAddC1213 + MddAddC1221 + MddAddC1222 + MddAddC1223 +
MddAddC1231 + MddAddC1232 + MddAddC1233);

% Calculation of the 1,3 Component of the MddAddC 2nd Order Tensor

MddAddC1311 = eigAvec2(1)^3             *eigAvec2(3)    *Explicit_IBOF_v(C,A,1,1);
MddAddC1312 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *Explicit_IBOF_v(C,A,2,1);
MddAddC1313 = eigAvec2(1)^2             *eigAvec2(3)^2   *Explicit_IBOF_v(C,A,3,1);
```

```
MddAddC1321 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *Explicit_IBOF_v(C,A,1,2);
MddAddC1322 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *Explicit_IBOF_v(C,A,2,2);
MddAddC1323 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *Explicit_IBOF_v(C,A,3,2);
MddAddC1331 = eigAvec2(1)^2              *eigAvec2(3)^2  *Explicit_IBOF_v(C,A,1,3);
MddAddC1332 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *Explicit_IBOF_v(C,A,2,3);
MddAddC1333 = eigAvec2(1)                *eigAvec2(3)^3  *Explicit_IBOF_v(C,A,3,3);

MddAddC13_e2 = (MddAddC1311 + MddAddC1312 + MddAddC1313 + MddAddC1321 + MddAddC1322 + MddAddC1323 +
MddAddC1331 + MddAddC1332 + MddAddC1333);

% Calculation of the 2,1 Component of the MddAddC 2nd Order Tensor

MddAddC21_e2 = MddAddC12_e2;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the MddAddC 2nd Order Tensor

MddAddC2211 = eigAvec2(1)^2 *eigAvec2(2)^2               *Explicit_IBOF_v(C,A,1,1);
MddAddC2212 = eigAvec2(1)   *eigAvec2(2)^3               *Explicit_IBOF_v(C,A,2,1);
MddAddC2213 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)  *Explicit_IBOF_v(C,A,3,1);
MddAddC2221 = eigAvec2(1)   *eigAvec2(2)^3               *Explicit_IBOF_v(C,A,1,2);
MddAddC2222 =               eigAvec2(2)^4                *Explicit_IBOF_v(C,A,2,2);
MddAddC2223 =               eigAvec2(2)^3 *eigAvec2(3)   *Explicit_IBOF_v(C,A,3,2);
MddAddC2231 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)  *Explicit_IBOF_v(C,A,1,3);
MddAddC2232 =               eigAvec2(2)^3 *eigAvec2(3)   *Explicit_IBOF_v(C,A,2,3);
MddAddC2233 =               eigAvec2(2)^2 *eigAvec2(3)^2 *Explicit_IBOF_v(C,A,3,3);

MddAddC22_e2 = (MddAddC2211 + MddAddC2212 + MddAddC2213 + MddAddC2221 + MddAddC2222 + MddAddC2223 +
MddAddC2231 + MddAddC2232 + MddAddC2233);

% Calculation of the 2,3 Component of the MddAddC 2nd Order Tensor

MddAddC2311 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *Explicit_IBOF_v(C,A,1,1);
MddAddC2312 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *Explicit_IBOF_v(C,A,2,1);
MddAddC2313 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *Explicit_IBOF_v(C,A,3,1);
MddAddC2321 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *Explicit_IBOF_v(C,A,1,2);
MddAddC2322 =               eigAvec2(2)^3 *eigAvec2(3)     *Explicit_IBOF_v(C,A,2,2);
MddAddC2323 =               eigAvec2(2)^2 *eigAvec2(3)^2   *Explicit_IBOF_v(C,A,3,2);
MddAddC2331 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *Explicit_IBOF_v(C,A,1,3);
MddAddC2332 =               eigAvec2(2)^2 *eigAvec2(3)^2   *Explicit_IBOF_v(C,A,2,3);
MddAddC2333 =               eigAvec2(2)   *eigAvec2(3)^3   *Explicit_IBOF_v(C,A,3,3);

MddAddC23_e2 = (MddAddC2311 + MddAddC2312 + MddAddC2313 + MddAddC2321 + MddAddC2322 + MddAddC2323 +
MddAddC2331 + MddAddC2332 + MddAddC2333);

% Calculation of the 3,1 Component of the MddAddC 2nd Order Tensor

MddAddC31_e2 = MddAddC13_e2;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the MddAddC 2nd Order Tensor

MddAddC32_e2 = MddAddC23_e2;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the MddAddC 2nd Order Tensor

MddAddC3311 = eigAvec2(1)^2              *eigAvec2(3)^2  *Explicit_IBOF_v(C,A,1,1);
MddAddC3312 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *Explicit_IBOF_v(C,A,2,1);
MddAddC3313 = eigAvec2(1)                *eigAvec2(3)^3  *Explicit_IBOF_v(C,A,3,1);
```

```
MddAddC3321 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2   *Explicit_IBOF_v(C,A,1,2);
MddAddC3322 =               eigAvec2(2)^2 *eigAvec2(3)^2   *Explicit_IBOF_v(C,A,2,2);
MddAddC3323 =               eigAvec2(2)   *eigAvec2(3)^3   *Explicit_IBOF_v(C,A,3,2);
MddAddC3331 = eigAvec2(1)                 *eigAvec2(3)^3   *Explicit_IBOF_v(C,A,1,3);
MddAddC3332 =               eigAvec2(2)   *eigAvec2(3)^3   *Explicit_IBOF_v(C,A,2,3);
MddAddC3333 =                             eigAvec2(3)^4    *Explicit_IBOF_v(C,A,3,3);

MddAddC33_e2 = (MddAddC3311 + MddAddC3312 + MddAddC3313 + MddAddC3321 + MddAddC3322 + MddAddC3323 +
MddAddC3331 + MddAddC3332 + MddAddC3333);

%% CALCULATION FOR I = 3

% Calculation of the 1,1 Component of the LddD 2nd Order Tensor

MddAddC1111 = eigAvec3(1)^4                           *Explicit_IBOF_v(C,A,1,1);
MddAddC1112 = eigAvec3(1)^3 *eigAvec3(2)              *Explicit_IBOF_v(C,A,2,1);
MddAddC1113 = eigAvec3(1)^3               *eigAvec3(3)*Explicit_IBOF_v(C,A,3,1);
MddAddC1121 = eigAvec3(1)^3 *eigAvec3(2)              *Explicit_IBOF_v(C,A,1,2);
MddAddC1122 = eigAvec3(1)^2 *eigAvec3(2)^2            *Explicit_IBOF_v(C,A,2,2);
MddAddC1123 = eigAvec3(1)^2 *eigAvec3(2)  *eigAvec3(3)*Explicit_IBOF_v(C,A,3,2);
MddAddC1131 = eigAvec3(1)^3               *eigAvec3(3)*Explicit_IBOF_v(C,A,1,3);
MddAddC1132 = eigAvec3(1)^2 *eigAvec3(2)  *eigAvec3(3)*Explicit_IBOF_v(C,A,2,3);
MddAddC1133 = eigAvec3(1)^2               *eigAvec3(3)^2 *Explicit_IBOF_v(C,A,3,3);

MddAddC11_e3 = (MddAddC1111 + MddAddC1112 + MddAddC1113 + MddAddC1121 + MddAddC1122 + MddAddC1123 +
MddAddC1131 + MddAddC1132 + MddAddC1133);

% Calculation of the 1,2 Component of the MddAddC 2nd Order Tensor

MddAddC1211 = eigAvec3(1)^3 *eigAvec3(2)              *Explicit_IBOF_v(C,A,1,1);
MddAddC1212 = eigAvec3(1)^2 *eigAvec3(2)^2            *Explicit_IBOF_v(C,A,2,1);
MddAddC1213 = eigAvec3(1)^2 *eigAvec3(2)  *eigAvec3(3)*Explicit_IBOF_v(C,A,3,1);
MddAddC1221 = eigAvec3(1)^2 *eigAvec3(2)^2            *Explicit_IBOF_v(C,A,1,2);
MddAddC1222 = eigAvec3(1)   *eigAvec3(2)^3            *Explicit_IBOF_v(C,A,2,2);
MddAddC1223 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)*Explicit_IBOF_v(C,A,3,2);
MddAddC1231 = eigAvec3(1)^2 *eigAvec3(2)  *eigAvec3(3)*Explicit_IBOF_v(C,A,1,3);
MddAddC1232 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)*Explicit_IBOF_v(C,A,2,3);
MddAddC1233 = eigAvec3(1)   *eigAvec3(2)  *eigAvec3(3)^2 *Explicit_IBOF_v(C,A,3,3);

MddAddC12_e3 = (MddAddC1211 + MddAddC1212 + MddAddC1213 + MddAddC1221 + MddAddC1222 + MddAddC1223 +
MddAddC1231 + MddAddC1232 + MddAddC1233);

% Calculation of the 1,3 Component of the MddAddC 2nd Order Tensor

MddAddC1311 = eigAvec3(1)^3               *eigAvec3(3)  *Explicit_IBOF_v(C,A,1,1);
MddAddC1312 = eigAvec3(1)^2 *eigAvec3(2)  *eigAvec3(3)  *Explicit_IBOF_v(C,A,2,1);
MddAddC1313 = eigAvec3(1)^2               *eigAvec3(3)^2 *Explicit_IBOF_v(C,A,3,1);
MddAddC1321 = eigAvec3(1)^2 *eigAvec3(2)  *eigAvec3(3)  *Explicit_IBOF_v(C,A,1,2);
MddAddC1322 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3) *Explicit_IBOF_v(C,A,2,2);
MddAddC1323 = eigAvec3(1)   *eigAvec3(2)  *eigAvec3(3)^2 *Explicit_IBOF_v(C,A,3,2);
MddAddC1331 = eigAvec3(1)^2               *eigAvec3(3)^2 *Explicit_IBOF_v(C,A,1,3);
MddAddC1332 = eigAvec3(1)   *eigAvec3(2)  *eigAvec3(3)^2 *Explicit_IBOF_v(C,A,2,3);
MddAddC1333 = eigAvec3(1)                 *eigAvec3(3)^3 *Explicit_IBOF_v(C,A,3,3);

MddAddC13_e3 = (MddAddC1311 + MddAddC1312 + MddAddC1313 + MddAddC1321 + MddAddC1322 + MddAddC1323 +
MddAddC1331 + MddAddC1332 + MddAddC1333);

% Calculation of the 2,1 Component of the MddAddC 2nd Order Tensor
```

```
MddAddC21_e3 = MddAddC12_e3;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the MddAddC 2nd Order Tensor

MddAddC2211 = eigAvec3(1)^2 *eigAvec3(2)^2              *Explicit_IBOF_v(C,A,1,1);
MddAddC2212 = eigAvec3(1)   *eigAvec3(2)^3              *Explicit_IBOF_v(C,A,2,1);
MddAddC2213 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *Explicit_IBOF_v(C,A,3,1);
MddAddC2221 = eigAvec3(1)   *eigAvec3(2)^3              *Explicit_IBOF_v(C,A,1,2);
MddAddC2222 =               eigAvec3(2)^4               *Explicit_IBOF_v(C,A,2,2);
MddAddC2223 =               eigAvec3(2)^3 *eigAvec3(3)    *Explicit_IBOF_v(C,A,3,2);
MddAddC2231 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *Explicit_IBOF_v(C,A,1,3);
MddAddC2232 =               eigAvec3(2)^3 *eigAvec3(3)    *Explicit_IBOF_v(C,A,2,3);
MddAddC2233 =               eigAvec3(2)^2 *eigAvec3(3)^2  *Explicit_IBOF_v(C,A,3,3);

MddAddC22_e3 = (MddAddC2211 + MddAddC2212 + MddAddC2213 + MddAddC2221 + MddAddC2222 + MddAddC2223 +
MddAddC2231 + MddAddC2232 + MddAddC2233);

% Calculation of the 2,3 Component of the MddAddC 2nd Order Tensor

MddAddC2311 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *Explicit_IBOF_v(C,A,1,1);
MddAddC2312 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *Explicit_IBOF_v(C,A,2,1);
MddAddC2313 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2  *Explicit_IBOF_v(C,A,3,1);
MddAddC2321 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *Explicit_IBOF_v(C,A,1,2);
MddAddC2322 =               eigAvec3(2)^3 *eigAvec3(3)    *Explicit_IBOF_v(C,A,2,2);
MddAddC2323 =               eigAvec3(2)^2 *eigAvec3(3)^2  *Explicit_IBOF_v(C,A,3,2);
MddAddC2331 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2  *Explicit_IBOF_v(C,A,1,3);
MddAddC2332 =               eigAvec3(2)^2 *eigAvec3(3)^2  *Explicit_IBOF_v(C,A,2,3);
MddAddC2333 =               eigAvec3(2)   *eigAvec3(3)^3  *Explicit_IBOF_v(C,A,3,3);

MddAddC23_e3 = (MddAddC2311 + MddAddC2312 + MddAddC2313 + MddAddC2321 + MddAddC2322 + MddAddC2323 +
MddAddC2331 + MddAddC2332 + MddAddC2333);

% Calculation of the 3,1 Component of the MddAddC 2nd Order Tensor

MddAddC31_e3 = MddAddC13_e3;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the MddAddC 2nd Order Tensor

MddAddC32_e3 = MddAddC23_e3;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the MddAddC 2nd Order Tensor

MddAddC3311 = eigAvec3(1)^2             *eigAvec3(3)^2  *Explicit_IBOF_v(C,A,1,1);
MddAddC3312 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2  *Explicit_IBOF_v(C,A,2,1);
MddAddC3313 = eigAvec3(1)               *eigAvec3(3)^3  *Explicit_IBOF_v(C,A,3,1);
MddAddC3321 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2  *Explicit_IBOF_v(C,A,1,2);
MddAddC3322 =               eigAvec3(2)^2 *eigAvec3(3)^2  *Explicit_IBOF_v(C,A,2,2);
MddAddC3323 =               eigAvec3(2)   *eigAvec3(3)^3  *Explicit_IBOF_v(C,A,3,2);
MddAddC3331 = eigAvec3(1)               *eigAvec3(3)^3  *Explicit_IBOF_v(C,A,1,3);
MddAddC3332 =               eigAvec3(2)   *eigAvec3(3)^3  *Explicit_IBOF_v(C,A,2,3);
MddAddC3333 =                           eigAvec3(3)^4   *Explicit_IBOF_v(C,A,3,3);

MddAddC33_e3 = (MddAddC3311 + MddAddC3312 + MddAddC3313 + MddAddC3321 + MddAddC3322 + MddAddC3323 +
MddAddC3331 + MddAddC3332 + MddAddC3333);

%% Summing the terms over i (i.e. i = 1,2,3)
```

```
MddAddC11 = MddAddC11_e1 + MddAddC11_e2 + MddAddC11_e3;
MddAddC12 = MddAddC12_e1 + MddAddC12_e2 + MddAddC12_e3;
MddAddC13 = MddAddC13_e1 + MddAddC13_e2 + MddAddC13_e3;
MddAddC21 = MddAddC21_e1 + MddAddC21_e2 + MddAddC21_e3;
MddAddC22 = MddAddC22_e1 + MddAddC22_e2 + MddAddC22_e3;
MddAddC23 = MddAddC23_e1 + MddAddC23_e2 + MddAddC23_e3;
MddAddC31 = MddAddC31_e1 + MddAddC31_e2 + MddAddC31_e3;
MddAddC32 = MddAddC32_e1 + MddAddC32_e2 + MddAddC32_e3;
MddAddC33 = MddAddC33_e1 + MddAddC33_e2 + MddAddC33_e3;

%% Forming the M:A:C Tensor (2nd Order)

MddAddC = [MddAddC11 MddAddC12 MddAddC13 MddAddC21 MddAddC22 MddAddC23 MddAddC31 MddAddC32
MddAddC33];
```

E.3.9 MddAddD.m

```
function [MddAddD] = MddAddD(D,A)

% Function written to compute the 4th order M tensor double dotted into the
% 4th order A tensor double dotted into the 2nd order rate of strain tensor (D).

% 4th Order A tensor used quadratic closure (AA)

[eigAvec,eigAval] = eig(reshape(A,3,3));   % Calculation of the numerical eigenvalues and eigenvectors.

eigAvec = fliplr(eigAvec);
eigAval = rot90(rot90(eigAval));

% Assigning the numerical values to discrete variables to ease in
% calculation.

eigAval1 = eigAval(1);
eigAval2 = eigAval(5);
eigAval3 = eigAval(9);

eigAvec1 = eigAvec(:,1);
eigAvec2 = eigAvec(:,2);
eigAvec3 = eigAvec(:,3);

%% CALCULATION FOR I = 1

% Calculation of the 1,1 Component of the LddD 2nd Order Tensor

MddAddD1111 = eigAvec1(1)^4                          *A(1);
MddAddD1112 = eigAvec1(1)^3 *eigAvec1(2)             *A(4);
MddAddD1113 = eigAvec1(1)^3             *eigAvec1(3)  *A(7);
MddAddD1121 = eigAvec1(1)^3 *eigAvec1(2)             *A(2);
MddAddD1122 = eigAvec1(1)^2 *eigAvec1(2)^2           *A(5);
MddAddD1123 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)  *A(8);
MddAddD1131 = eigAvec1(1)^3             *eigAvec1(3)  *A(3);
MddAddD1132 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)  *A(6);
MddAddD1133 = eigAvec1(1)^2             *eigAvec1(3)^2  *A(9);

MddAddD11_e1 = Explicit_Quad_v(D,A)*(MddAddD1111 + MddAddD1112 + MddAddD1113 + MddAddD1121 +
MddAddD1122 + MddAddD1123 + MddAddD1131 + MddAddD1132 + MddAddD1133);
```

```
% Calculation of the 1,2 Component of the MddAddD 2nd Order Tensor

MddAddD1211 = eigAvec1(1)^3 *eigAvec1(2)                *A(1);
MddAddD1212 = eigAvec1(1)^2 *eigAvec1(2)^2              *A(4);
MddAddD1213 = eigAvec1(1)^2 *eigAvec1(2)   *eigAvec1(3)    *A(7);
MddAddD1221 = eigAvec1(1)^2 *eigAvec1(2)^2              *A(2);
MddAddD1222 = eigAvec1(1)   *eigAvec1(2)^3              *A(5);
MddAddD1223 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)    *A(8);
MddAddD1231 = eigAvec1(1)^2 *eigAvec1(2)   *eigAvec1(3)    *A(3);
MddAddD1232 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)    *A(6);
MddAddD1233 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2  *A(9);

MddAddD12_e1 = Explicit_Quad_v(D,A)*(MddAddD1211 + MddAddD1212 + MddAddD1213 + MddAddD1221 +
MddAddD1222 + MddAddD1223 + MddAddD1231 + MddAddD1232 + MddAddD1233);

% Calculation of the 1,3 Component of the MddAddD 2nd Order Tensor

MddAddD1311 = eigAvec1(1)^3             *eigAvec1(3)    *A(1);
MddAddD1312 = eigAvec1(1)^2 *eigAvec1(2)   *eigAvec1(3)    *A(4);
MddAddD1313 = eigAvec1(1)^2             *eigAvec1(3)^2  *A(7);
MddAddD1321 = eigAvec1(1)^2 *eigAvec1(2)   *eigAvec1(3)    *A(2);
MddAddD1322 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)    *A(5);
MddAddD1323 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2  *A(8);
MddAddD1331 = eigAvec1(1)^2             *eigAvec1(3)^2  *A(3);
MddAddD1332 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2  *A(6);
MddAddD1333 = eigAvec1(1)               *eigAvec1(3)^3  *A(9);

MddAddD13_e1 = Explicit_Quad_v(D,A)*(MddAddD1311 + MddAddD1312 + MddAddD1313 + MddAddD1321 +
MddAddD1322 + MddAddD1323 + MddAddD1331 + MddAddD1332 + MddAddD1333);

% Calculation of the 2,1 Component of the MddAddD 2nd Order Tensor

MddAddD21_e1 = MddAddD12_e1;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the MddAddD 2nd Order Tensor

MddAddD2211 = eigAvec1(1)^2 *eigAvec1(2)^2              *A(1);
MddAddD2212 = eigAvec1(1)   *eigAvec1(2)^3              *A(4);
MddAddD2213 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)    *A(7);
MddAddD2221 = eigAvec1(1)   *eigAvec1(2)^3              *A(2);
MddAddD2222 =               eigAvec1(2)^4               *A(5);
MddAddD2223 =               eigAvec1(2)^3 *eigAvec1(3)    *A(8);
MddAddD2231 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)    *A(3);
MddAddD2232 =               eigAvec1(2)^3 *eigAvec1(3)    *A(6);
MddAddD2233 =               eigAvec1(2)^2 *eigAvec1(3)^2  *A(9);

MddAddD22_e1 = Explicit_Quad_v(D,A)*(MddAddD2211 + MddAddD2212 + MddAddD2213 + MddAddD2221 +
MddAddD2222 + MddAddD2223 + MddAddD2231 + MddAddD2232 + MddAddD2233);

% Calculation of the 2,3 Component of the MddAddD 2nd Order Tensor

MddAddD2311 = eigAvec1(1)^2 *eigAvec1(2)   *eigAvec1(3)    *A(1);
MddAddD2312 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)    *A(4);
MddAddD2313 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2  *A(7);
MddAddD2321 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)    *A(2);
MddAddD2322 =               eigAvec1(2)^3 *eigAvec1(3)    *A(5);
```

```
MddAddD2323 =              eigAvec1(2)^2 *eigAvec1(3)^2   *A(8);
MddAddD2331 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *A(3);
MddAddD2332 =              eigAvec1(2)^2 *eigAvec1(3)^2   *A(6);
MddAddD2333 =              eigAvec1(2)   *eigAvec1(3)^3   *A(9);

MddAddD23_e1 = Explicit_Quad_v(D,A)*(MddAddD2311 + MddAddD2312 + MddAddD2313 + MddAddD2321 +
MddAddD2322 + MddAddD2323 + MddAddD2331 + MddAddD2332 + MddAddD2333);

% Calculation of the 3,1 Component of the MddAddD 2nd Order Tensor

MddAddD31_e1 = MddAddD13_e1;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the MddAddD 2nd Order Tensor

MddAddD32_e1 = MddAddD23_e1;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the MddAddD 2nd Order Tensor

MddAddD3311 = eigAvec1(1)^2              *eigAvec1(3)^2   *A(1);
MddAddD3312 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *A(4);
MddAddD3313 = eigAvec1(1)              *eigAvec1(3)^3   *A(7);
MddAddD3321 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *A(2);
MddAddD3322 =              eigAvec1(2)^2 *eigAvec1(3)^2   *A(5);
MddAddD3323 =              eigAvec1(2)   *eigAvec1(3)^3   *A(8);
MddAddD3331 = eigAvec1(1)              *eigAvec1(3)^3   *A(3);
MddAddD3332 =              eigAvec1(2)   *eigAvec1(3)^3   *A(6);
MddAddD3333 =                        eigAvec1(3)^4   *A(9);

MddAddD33_e1 = Explicit_Quad_v(D,A)*(MddAddD3311 + MddAddD3312 + MddAddD3313 + MddAddD3321 +
MddAddD3322 + MddAddD3323 + MddAddD3331 + MddAddD3332 + MddAddD3333);

%% CALCULATION FOR I = 2

% Calculation of the 1,1 Component of the LddD 2nd Order Tensor

MddAddD1111 = eigAvec2(1)^4                        *A(1);
MddAddD1112 = eigAvec2(1)^3 *eigAvec2(2)              *A(4);
MddAddD1113 = eigAvec2(1)^3              *eigAvec2(3)    *A(7);
MddAddD1121 = eigAvec2(1)^3 *eigAvec2(2)              *A(2);
MddAddD1122 = eigAvec2(1)^2 *eigAvec2(2)^2             *A(5);
MddAddD1123 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *A(8);
MddAddD1131 = eigAvec2(1)^3              *eigAvec2(3)    *A(3);
MddAddD1132 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *A(6);
MddAddD1133 = eigAvec2(1)^2              *eigAvec2(3)^2  *A(9);

MddAddD11_e2 = Explicit_Quad_v(D,A)*(MddAddD1111 + MddAddD1112 + MddAddD1113 + MddAddD1121 +
MddAddD1122 + MddAddD1123 + MddAddD1131 + MddAddD1132 + MddAddD1133);

% Calculation of the 1,2 Component of the MddAddD 2nd Order Tensor

MddAddD1211 = eigAvec2(1)^3 *eigAvec2(2)              *A(1);
MddAddD1212 = eigAvec2(1)^2 *eigAvec2(2)^2             *A(4);
MddAddD1213 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *A(7);
MddAddD1221 = eigAvec2(1)^2 *eigAvec2(2)^2             *A(2);
MddAddD1222 = eigAvec2(1)   *eigAvec2(2)^3             *A(5);
MddAddD1223 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *A(8);
MddAddD1231 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *A(3);
```

```
MddAddD1232 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *A(6);
MddAddD1233 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *A(9);

MddAddD12_e2 = Explicit_Quad_v(D,A)*(MddAddD1211 + MddAddD1212 + MddAddD1213 + MddAddD1221 +
MddAddD1222 + MddAddD1223 + MddAddD1231 + MddAddD1232 + MddAddD1233);

% Calculation of the 1,3 Component of the MddAddD 2nd Order Tensor

MddAddD1311 = eigAvec2(1)^3              *eigAvec2(3)     *A(1);
MddAddD1312 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)     *A(4);
MddAddD1313 = eigAvec2(1)^2              *eigAvec2(3)^2   *A(7);
MddAddD1321 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)     *A(2);
MddAddD1322 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)     *A(5);
MddAddD1323 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2   *A(8);
MddAddD1331 = eigAvec2(1)^2              *eigAvec2(3)^2   *A(3);
MddAddD1332 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2   *A(6);
MddAddD1333 = eigAvec2(1)                *eigAvec2(3)^3   *A(9);

MddAddD13_e2 = Explicit_Quad_v(D,A)*(MddAddD1311 + MddAddD1312 + MddAddD1313 + MddAddD1321 +
MddAddD1322 + MddAddD1323 + MddAddD1331 + MddAddD1332 + MddAddD1333);

% Calculation of the 2,1 Component of the MddAddD 2nd Order Tensor

MddAddD21_e2 = MddAddD12_e2;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the MddAddD 2nd Order Tensor

MddAddD2211 = eigAvec2(1)^2 *eigAvec2(2)^2               *A(1);
MddAddD2212 = eigAvec2(1)   *eigAvec2(2)^3               *A(4);
MddAddD2213 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)     *A(7);
MddAddD2221 = eigAvec2(1)   *eigAvec2(2)^3               *A(2);
MddAddD2222 =               eigAvec2(2)^4                *A(5);
MddAddD2223 =               eigAvec2(2)^3 *eigAvec2(3)     *A(8);
MddAddD2231 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)     *A(3);
MddAddD2232 =               eigAvec2(2)^3 *eigAvec2(3)     *A(6);
MddAddD2233 =               eigAvec2(2)^2 *eigAvec2(3)^2   *A(9);

MddAddD22_e2 = Explicit_Quad_v(D,A)*(MddAddD2211 + MddAddD2212 + MddAddD2213 + MddAddD2221 +
MddAddD2222 + MddAddD2223 + MddAddD2231 + MddAddD2232 + MddAddD2233);

% Calculation of the 2,3 Component of the MddAddD 2nd Order Tensor

MddAddD2311 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)     *A(1);
MddAddD2312 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)     *A(4);
MddAddD2313 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2   *A(7);
MddAddD2321 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)     *A(2);
MddAddD2322 =               eigAvec2(2)^3 *eigAvec2(3)     *A(5);
MddAddD2323 =               eigAvec2(2)^2 *eigAvec2(3)^2   *A(8);
MddAddD2331 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2   *A(3);
MddAddD2332 =               eigAvec2(2)^2 *eigAvec2(3)^2   *A(6);
MddAddD2333 =               eigAvec2(2)   *eigAvec2(3)^3   *A(9);

MddAddD23_e2 = Explicit_Quad_v(D,A)*(MddAddD2311 + MddAddD2312 + MddAddD2313 + MddAddD2321 +
MddAddD2322 + MddAddD2323 + MddAddD2331 + MddAddD2332 + MddAddD2333);

% Calculation of the 3,1 Component of the MddAddD 2nd Order Tensor
```

```
MddAddD31_e2 = MddAddD13_e2;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the MddAddD 2nd Order Tensor

MddAddD32_e2 = MddAddD23_e2;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the MddAddD 2nd Order Tensor

MddAddD3311 = eigAvec2(1)^2              *eigAvec2(3)^2   *A(1);
MddAddD3312 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2   *A(4);
MddAddD3313 = eigAvec2(1)                *eigAvec2(3)^3   *A(7);
MddAddD3321 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2   *A(2);
MddAddD3322 =              eigAvec2(2)^2 *eigAvec2(3)^2   *A(5);
MddAddD3323 =              eigAvec2(2)   *eigAvec2(3)^3   *A(8);
MddAddD3331 = eigAvec2(1)                *eigAvec2(3)^3   *A(3);
MddAddD3332 =              eigAvec2(2)   *eigAvec2(3)^3   *A(6);
MddAddD3333 =                          eigAvec2(3)^4   *A(9);

MddAddD33_e2 = Explicit_Quad_v(D,A)*(MddAddD3311 + MddAddD3312 + MddAddD3313 + MddAddD3321 +
MddAddD3322 + MddAddD3323 + MddAddD3331 + MddAddD3332 + MddAddD3333);

%% CALCULATION FOR I = 3

% Calculation of the 1,1 Component of the LddD 2nd Order Tensor

MddAddD1111 = eigAvec3(1)^4                          *A(1);
MddAddD1112 = eigAvec3(1)^3 *eigAvec3(2)               *A(4);
MddAddD1113 = eigAvec3(1)^3              *eigAvec3(3)    *A(7);
MddAddD1121 = eigAvec3(1)^3 *eigAvec3(2)               *A(2);
MddAddD1122 = eigAvec3(1)^2 *eigAvec3(2)^2              *A(5);
MddAddD1123 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *A(8);
MddAddD1131 = eigAvec3(1)^3              *eigAvec3(3)    *A(3);
MddAddD1132 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *A(6);
MddAddD1133 = eigAvec3(1)^2              *eigAvec3(3)^2  *A(9);

MddAddD11_e3 = Explicit_Quad_v(D,A)*(MddAddD1111 + MddAddD1112 + MddAddD1113 + MddAddD1121 +
MddAddD1122 + MddAddD1123 + MddAddD1131 + MddAddD1132 + MddAddD1133);

% Calculation of the 1,2 Component of the MddAddD 2nd Order Tensor

MddAddD1211 = eigAvec3(1)^3 *eigAvec3(2)               *A(1);
MddAddD1212 = eigAvec3(1)^2 *eigAvec3(2)^2              *A(4);
MddAddD1213 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *A(7);
MddAddD1221 = eigAvec3(1)^2 *eigAvec3(2)^2              *A(2);
MddAddD1222 = eigAvec3(1)   *eigAvec3(2)^3              *A(5);
MddAddD1223 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *A(8);
MddAddD1231 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *A(3);
MddAddD1232 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *A(6);
MddAddD1233 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2  *A(9);

MddAddD12_e3 = Explicit_Quad_v(D,A)*(MddAddD1211 + MddAddD1212 + MddAddD1213 + MddAddD1221 +
MddAddD1222 + MddAddD1223 + MddAddD1231 + MddAddD1232 + MddAddD1233);

% Calculation of the 1,3 Component of the MddAddD 2nd Order Tensor

MddAddD1311 = eigAvec3(1)^3              *eigAvec3(3)    *A(1);
MddAddD1312 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *A(4);
```

```
MddAddD1313 = eigAvec3(1)^2              *eigAvec3(3)^2   *A(7);
MddAddD1321 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)     *A(2);
MddAddD1322 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)     *A(5);
MddAddD1323 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *A(8);
MddAddD1331 = eigAvec3(1)^2              *eigAvec3(3)^2   *A(3);
MddAddD1332 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *A(6);
MddAddD1333 = eigAvec3(1)                *eigAvec3(3)^3   *A(9);

MddAddD13_e3 = Explicit_Quad_v(D,A)*(MddAddD1311 + MddAddD1312 + MddAddD1313 + MddAddD1321 +
MddAddD1322 + MddAddD1323 + MddAddD1331 + MddAddD1332 + MddAddD1333);

% Calculation of the 2,1 Component of the MddAddD 2nd Order Tensor

MddAddD21_e3 = MddAddD12_e3;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the MddAddD 2nd Order Tensor

MddAddD2211 = eigAvec3(1)^2 *eigAvec3(2)^2               *A(1);
MddAddD2212 = eigAvec3(1)   *eigAvec3(2)^3               *A(4);
MddAddD2213 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)     *A(7);
MddAddD2221 = eigAvec3(1)   *eigAvec3(2)^3               *A(2);
MddAddD2222 =               eigAvec3(2)^4                *A(5);
MddAddD2223 =               eigAvec3(2)^3 *eigAvec3(3)     *A(8);
MddAddD2231 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)     *A(3);
MddAddD2232 =               eigAvec3(2)^3 *eigAvec3(3)     *A(6);
MddAddD2233 =               eigAvec3(2)^2 *eigAvec3(3)^2   *A(9);

MddAddD22_e3 = Explicit_Quad_v(D,A)*(MddAddD2211 + MddAddD2212 + MddAddD2213 + MddAddD2221 +
MddAddD2222 + MddAddD2223 + MddAddD2231 + MddAddD2232 + MddAddD2233);

% Calculation of the 2,3 Component of the MddAddD 2nd Order Tensor

MddAddD2311 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)     *A(1);
MddAddD2312 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)     *A(4);
MddAddD2313 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *A(7);
MddAddD2321 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)     *A(2);
MddAddD2322 =               eigAvec3(2)^3 *eigAvec3(3)     *A(5);
MddAddD2323 =               eigAvec3(2)^2 *eigAvec3(3)^2   *A(8);
MddAddD2331 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *A(3);
MddAddD2332 =               eigAvec3(2)^2 *eigAvec3(3)^2   *A(6);
MddAddD2333 =               eigAvec3(2)   *eigAvec3(3)^3   *A(9);

MddAddD23_e3 = Explicit_Quad_v(D,A)*(MddAddD2311 + MddAddD2312 + MddAddD2313 + MddAddD2321 +
MddAddD2322 + MddAddD2323 + MddAddD2331 + MddAddD2332 + MddAddD2333);

% Calculation of the 3,1 Component of the MddAddD 2nd Order Tensor

MddAddD31_e3 = MddAddD13_e3;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the MddAddD 2nd Order Tensor

MddAddD32_e3 = MddAddD23_e3;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the MddAddD 2nd Order Tensor

MddAddD3311 = eigAvec3(1)^2              *eigAvec3(3)^2   *A(1);
MddAddD3312 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *A(4);
```

```
MddAddD3313 = eigAvec3(1)               *eigAvec3(3)^3   *A(7);
MddAddD3321 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *A(2);
MddAddD3322 =             eigAvec3(2)^2 *eigAvec3(3)^2   *A(5);
MddAddD3323 =             eigAvec3(2)   *eigAvec3(3)^3   *A(8);
MddAddD3331 = eigAvec3(1)               *eigAvec3(3)^3   *A(3);
MddAddD3332 =             eigAvec3(2)   *eigAvec3(3)^3   *A(6);
MddAddD3333 =                         eigAvec3(3)^4   *A(9);

MddAddD33_e3 = Explicit_Quad_v(D,A)*(MddAddD3311 + MddAddD3312 + MddAddD3313 + MddAddD3321 +
MddAddD3322 + MddAddD3323 + MddAddD3331 + MddAddD3332 + MddAddD3333);

%% Summing the terms over i (i.e. i = 1,2,3)

MddAddD11 = MddAddD11_e1 + MddAddD11_e2 + MddAddD11_e3;
MddAddD12 = MddAddD12_e1 + MddAddD12_e2 + MddAddD12_e3;
MddAddD13 = MddAddD13_e1 + MddAddD13_e2 + MddAddD13_e3;
MddAddD21 = MddAddD21_e1 + MddAddD21_e2 + MddAddD21_e3;
MddAddD22 = MddAddD22_e1 + MddAddD22_e2 + MddAddD22_e3;
MddAddD23 = MddAddD23_e1 + MddAddD23_e2 + MddAddD23_e3;
MddAddD31 = MddAddD32_e1 + MddAddD31_e2 + MddAddD31_e3;
MddAddD32 = MddAddD32_e1 + MddAddD32_e2 + MddAddD32_e3;
MddAddD33 = MddAddD33_e1 + MddAddD33_e2 + MddAddD33_e3;

%% Forming the M:A:D Tensor (M:A:D)

MddAddD = [MddAddD11 MddAddD12 MddAddD13 MddAddD21 MddAddD22 MddAddD23 MddAddD31 MddAddD32
MddAddD33];
%
% MddAddD(isnan(MddAddD)) = 0;
% MddAddD(isinf(MddAddD)) = 0;
```

E.3.10 MddAddD_IBOF.m

```
function [MddAddD] = MddAddD_IBOF(D,A)

% Function written to compute the 4th order M tensor double dotted into the
% 4th Order A tensor double dotted into 2nd order rate of strain tensor (D).

% 4th Order A Tensor Used IBOF Closure Approximation.

[eigAvec,eigAval] = eig(reshape(A,3,3));   % Calculation of the numerical eigenvalues and eigenvectors.

eigAvec = fliplr(eigAvec);
eigAval = rot90(rot90(eigAval));

% Assigning the numerical values to discrete variables to ease in
% calculation.

eigAvec1 = eigAvec(:,1);
eigAvec2 = eigAvec(:,2);
eigAvec3 = eigAvec(:,3);

A11IBOF_val = Explicit_IBOF_v(D,A,1,1);
A12IBOF_val = Explicit_IBOF_v(D,A,1,2);
A13IBOF_val = Explicit_IBOF_v(D,A,1,3);
A22IBOF_val = Explicit_IBOF_v(D,A,2,2);
A23IBOF_val = Explicit_IBOF_v(D,A,2,3);
```

```
A33IBOF_val = Explicit_IBOF_v(D,A,3,3);

%% CALCULATION FOR I = 1

% Calculation of the 1,1 Component of the LddD 2nd Order Tensor

MddAddD1111 = eigAvec1(1)^4                         *A11IBOF_val;
MddAddD1112 = eigAvec1(1)^3 *eigAvec1(2)            *A12IBOF_val;
MddAddD1113 = eigAvec1(1)^3          *eigAvec1(3)   *A13IBOF_val;
MddAddD1121 = eigAvec1(1)^3 *eigAvec1(2)            *A12IBOF_val;
MddAddD1122 = eigAvec1(1)^2 *eigAvec1(2)^2          *A22IBOF_val;
MddAddD1123 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)   *A23IBOF_val;
MddAddD1131 = eigAvec1(1)^3          *eigAvec1(3)   *A13IBOF_val;
MddAddD1132 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)   *A23IBOF_val;
MddAddD1133 = eigAvec1(1)^2          *eigAvec1(3)^2 *A33IBOF_val;

MddAddD11_e1 = (MddAddD1111 + MddAddD1112 + MddAddD1113 + MddAddD1121 + MddAddD1122 + MddAddD1123
+ MddAddD1131 + MddAddD1132 + MddAddD1133);

% Calculation of the 1,2 Component of the MddAddD 2nd Order Tensor

MddAddD1211 = eigAvec1(1)^3 *eigAvec1(2)            *A11IBOF_val;
MddAddD1212 = eigAvec1(1)^2 *eigAvec1(2)^2          *A12IBOF_val;
MddAddD1213 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)   *A13IBOF_val;
MddAddD1221 = eigAvec1(1)^2 *eigAvec1(2)^2          *A12IBOF_val;
MddAddD1222 = eigAvec1(1)  *eigAvec1(2)^3           *A22IBOF_val;
MddAddD1223 = eigAvec1(1)  *eigAvec1(2)^2 *eigAvec1(3)   *A23IBOF_val;
MddAddD1231 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)   *A13IBOF_val;
MddAddD1232 = eigAvec1(1)  *eigAvec1(2)^2 *eigAvec1(3)   *A23IBOF_val;
MddAddD1233 = eigAvec1(1)  *eigAvec1(2)  *eigAvec1(3)^2 *A33IBOF_val;

MddAddD12_e1 = (MddAddD1211 + MddAddD1212 + MddAddD1213 + MddAddD1221 + MddAddD1222 + MddAddD1223
+ MddAddD1231 + MddAddD1232 + MddAddD1233);

% Calculation of the 1,3 Component of the MddAddD 2nd Order Tensor

MddAddD1311 = eigAvec1(1)^3          *eigAvec1(3)   *A11IBOF_val;
MddAddD1312 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)   *A12IBOF_val;
MddAddD1313 = eigAvec1(1)^2          *eigAvec1(3)^2 *A13IBOF_val;
MddAddD1321 = eigAvec1(1)^2 *eigAvec1(2)  *eigAvec1(3)   *A12IBOF_val;
MddAddD1322 = eigAvec1(1)  *eigAvec1(2)^2 *eigAvec1(3)   *A22IBOF_val;
MddAddD1323 = eigAvec1(1)  *eigAvec1(2)  *eigAvec1(3)^2 *A23IBOF_val;
MddAddD1331 = eigAvec1(1)^2          *eigAvec1(3)^2 *A13IBOF_val;
MddAddD1332 = eigAvec1(1)  *eigAvec1(2)  *eigAvec1(3)^2 *A23IBOF_val;
MddAddD1333 = eigAvec1(1)           *eigAvec1(3)^3 *A33IBOF_val;

MddAddD13_e1 = (MddAddD1311 + MddAddD1312 + MddAddD1313 + MddAddD1321 + MddAddD1322 + MddAddD1323
+ MddAddD1331 + MddAddD1332 + MddAddD1333);

% Calculation of the 2,1 Component of the MddAddD 2nd Order Tensor

MddAddD21_e1 = MddAddD12_e1;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the MddAddD 2nd Order Tensor

MddAddD2211 = eigAvec1(1)^2 *eigAvec1(2)^2          *A11IBOF_val;
MddAddD2212 = eigAvec1(1)  *eigAvec1(2)^3           *A12IBOF_val;
```

```
MddAddD2213 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)     *A13IBOF_val;
MddAddD2221 = eigAvec1(1)   *eigAvec1(2)^3               *A12IBOF_val;
MddAddD2222 =               eigAvec1(2)^4               *A22IBOF_val;
MddAddD2223 =               eigAvec1(2)^3 *eigAvec1(3)     *A23IBOF_val;
MddAddD2231 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)     *A13IBOF_val;
MddAddD2232 =               eigAvec1(2)^3 *eigAvec1(3)     *A23IBOF_val;
MddAddD2233 =               eigAvec1(2)^2 *eigAvec1(3)^2   *A33IBOF_val;

MddAddD22_e1 = (MddAddD2211 + MddAddD2212 + MddAddD2213 + MddAddD2221 + MddAddD2222 + MddAddD2223
+ MddAddD2231 + MddAddD2232 + MddAddD2233);

% Calculation of the 2,3 Component of the MddAddD 2nd Order Tensor

MddAddD2311 = eigAvec1(1)^2 *eigAvec1(2)   *eigAvec1(3)     *A11IBOF_val;
MddAddD2312 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)     *A12IBOF_val;
MddAddD2313 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *A13IBOF_val;
MddAddD2321 = eigAvec1(1)   *eigAvec1(2)^2 *eigAvec1(3)     *A12IBOF_val;
MddAddD2322 =               eigAvec1(2)^3 *eigAvec1(3)     *A22IBOF_val;
MddAddD2323 =               eigAvec1(2)^2 *eigAvec1(3)^2   *A23IBOF_val;
MddAddD2331 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *A13IBOF_val;
MddAddD2332 =               eigAvec1(2)^2 *eigAvec1(3)^2   *A23IBOF_val;
MddAddD2333 =               eigAvec1(2)   *eigAvec1(3)^3   *A33IBOF_val;

MddAddD23_e1 = (MddAddD2311 + MddAddD2312 + MddAddD2313 + MddAddD2321 + MddAddD2322 + MddAddD2323
+ MddAddD2331 + MddAddD2332 + MddAddD2333);

% Calculation of the 3,1 Component of the MddAddD 2nd Order Tensor

MddAddD31_e1 = MddAddD13_e1;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the MddAddD 2nd Order Tensor

MddAddD32_e1 = MddAddD23_e1;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the MddAddD 2nd Order Tensor

MddAddD3311 = eigAvec1(1)^2               *eigAvec1(3)^2   *A11IBOF_val;
MddAddD3312 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *A12IBOF_val;
MddAddD3313 = eigAvec1(1)                 *eigAvec1(3)^3   *A13IBOF_val;
MddAddD3321 = eigAvec1(1)   *eigAvec1(2)   *eigAvec1(3)^2   *A12IBOF_val;
MddAddD3322 =               eigAvec1(2)^2 *eigAvec1(3)^2   *A22IBOF_val;
MddAddD3323 =               eigAvec1(2)   *eigAvec1(3)^3   *A23IBOF_val;
MddAddD3331 = eigAvec1(1)                 *eigAvec1(3)^3   *A13IBOF_val;
MddAddD3332 =               eigAvec1(2)   *eigAvec1(3)^3   *A23IBOF_val;
MddAddD3333 =                             eigAvec1(3)^4    *A33IBOF_val;

MddAddD33_e1 = (MddAddD3311 + MddAddD3312 + MddAddD3313 + MddAddD3321 + MddAddD3322 + MddAddD3323
+ MddAddD3331 + MddAddD3332 + MddAddD3333);

%% CALCULATION FOR I = 2

% Calculation of the 1,1 Component of the LddD 2nd Order Tensor

MddAddD1111 = eigAvec2(1)^4                         *A11IBOF_val;
MddAddD1112 = eigAvec2(1)^3 *eigAvec2(2)               *A12IBOF_val;
MddAddD1113 = eigAvec2(1)^3               *eigAvec2(3)    *A13IBOF_val;
MddAddD1121 = eigAvec2(1)^3 *eigAvec2(2)               *A12IBOF_val;
```

```
MddAddD1122 = eigAvec2(1)^2 *eigAvec2(2)^2              *A22IBOF_val;
MddAddD1123 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *A23IBOF_val;
MddAddD1131 = eigAvec2(1)^3              *eigAvec2(3)    *A13IBOF_val;
MddAddD1132 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *A23IBOF_val;
MddAddD1133 = eigAvec2(1)^2              *eigAvec2(3)^2  *A33IBOF_val;

MddAddD11_e2 = (MddAddD1111 + MddAddD1112 + MddAddD1113 + MddAddD1121 + MddAddD1122 + MddAddD1123
+ MddAddD1131 + MddAddD1132 + MddAddD1133);

% Calculation of the 1,2 Component of the MddAddD 2nd Order Tensor

MddAddD1211 = eigAvec2(1)^3 *eigAvec2(2)                *A11IBOF_val;
MddAddD1212 = eigAvec2(1)^2 *eigAvec2(2)^2              *A12IBOF_val;
MddAddD1213 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *A13IBOF_val;
MddAddD1221 = eigAvec2(1)^2 *eigAvec2(2)^2              *A12IBOF_val;
MddAddD1222 = eigAvec2(1)   *eigAvec2(2)^3              *A22IBOF_val;
MddAddD1223 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *A23IBOF_val;
MddAddD1231 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *A13IBOF_val;
MddAddD1232 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *A23IBOF_val;
MddAddD1233 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *A33IBOF_val;

MddAddD12_e2 = (MddAddD1211 + MddAddD1212 + MddAddD1213 + MddAddD1221 + MddAddD1222 + MddAddD1223
+ MddAddD1231 + MddAddD1232 + MddAddD1233);

% Calculation of the 1,3 Component of the MddAddD 2nd Order Tensor

MddAddD1311 = eigAvec2(1)^3              *eigAvec2(3)    *A11IBOF_val;
MddAddD1312 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *A12IBOF_val;
MddAddD1313 = eigAvec2(1)^2              *eigAvec2(3)^2  *A13IBOF_val;
MddAddD1321 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *A12IBOF_val;
MddAddD1322 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *A22IBOF_val;
MddAddD1323 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *A23IBOF_val;
MddAddD1331 = eigAvec2(1)^2              *eigAvec2(3)^2  *A13IBOF_val;
MddAddD1332 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *A23IBOF_val;
MddAddD1333 = eigAvec2(1)                *eigAvec2(3)^3  *A33IBOF_val;

MddAddD13_e2 = (MddAddD1311 + MddAddD1312 + MddAddD1313 + MddAddD1321 + MddAddD1322 + MddAddD1323
+ MddAddD1331 + MddAddD1332 + MddAddD1333);

% Calculation of the 2,1 Component of the MddAddD 2nd Order Tensor

MddAddD21_e2 = MddAddD12_e2;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the MddAddD 2nd Order Tensor

MddAddD2211 = eigAvec2(1)^2 *eigAvec2(2)^2              *A11IBOF_val;
MddAddD2212 = eigAvec2(1)   *eigAvec2(2)^3              *A12IBOF_val;
MddAddD2213 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *A13IBOF_val;
MddAddD2221 = eigAvec2(1)   *eigAvec2(2)^3              *A12IBOF_val;
MddAddD2222 =               eigAvec2(2)^4               *A22IBOF_val;
MddAddD2223 =               eigAvec2(2)^3 *eigAvec2(3)    *A23IBOF_val;
MddAddD2231 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *A13IBOF_val;
MddAddD2232 =               eigAvec2(2)^3 *eigAvec2(3)    *A23IBOF_val;
MddAddD2233 =               eigAvec2(2)^2 *eigAvec2(3)^2  *A33IBOF_val;

MddAddD22_e2 = (MddAddD2211 + MddAddD2212 + MddAddD2213 + MddAddD2221 + MddAddD2222 + MddAddD2223
+ MddAddD2231 + MddAddD2232 + MddAddD2233);
```

```
% Calculation of the 2,3 Component of the MddAddD 2nd Order Tensor

MddAddD2311 = eigAvec2(1)^2 *eigAvec2(2)   *eigAvec2(3)    *A11IBOF_val;
MddAddD2312 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *A12IBOF_val;
MddAddD2313 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *A13IBOF_val;
MddAddD2321 = eigAvec2(1)   *eigAvec2(2)^2 *eigAvec2(3)    *A12IBOF_val;
MddAddD2322 =               eigAvec2(2)^3 *eigAvec2(3)     *A22IBOF_val;
MddAddD2323 =               eigAvec2(2)^2 *eigAvec2(3)^2   *A23IBOF_val;
MddAddD2331 = eigAvec2(1)   *eigAvec2(2)   *eigAvec2(3)^2  *A13IBOF_val;
MddAddD2332 =               eigAvec2(2)^2 *eigAvec2(3)^2   *A23IBOF_val;
MddAddD2333 =               eigAvec2(2)   *eigAvec2(3)^3   *A33IBOF_val;

MddAddD23_e2 = (MddAddD2311 + MddAddD2312 + MddAddD2313 + MddAddD2321 + MddAddD2322 + MddAddD2323
+ MddAddD2331 + MddAddD2332 + MddAddD2333);

% Calculation of the 3,1 Component of the MddAddD 2nd Order Tensor

MddAddD31_e2 = MddAddD13_e2;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the MddAddD 2nd Order Tensor

MddAddD32_e2 = MddAddD23_e2;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the MddAddD 2nd Order Tensor

MddAddD3311 = eigAvec2(1)^2               *eigAvec2(3)^2   *A11IBOF_val;
MddAddD3312 = eigAvec2(1)   *eigAvec2(2)  *eigAvec2(3)^2   *A12IBOF_val;
MddAddD3313 = eigAvec2(1)                 *eigAvec2(3)^3   *A13IBOF_val;
MddAddD3321 = eigAvec2(1)   *eigAvec2(2)  *eigAvec2(3)^2   *A12IBOF_val;
MddAddD3322 =               eigAvec2(2)^2 *eigAvec2(3)^2   *A22IBOF_val;
MddAddD3323 =               eigAvec2(2)   *eigAvec2(3)^3   *A23IBOF_val;
MddAddD3331 = eigAvec2(1)                 *eigAvec2(3)^3   *A13IBOF_val;
MddAddD3332 =               eigAvec2(2)   *eigAvec2(3)^3   *A23IBOF_val;
MddAddD3333 =                             eigAvec2(3)^4    *A33IBOF_val;

MddAddD33_e2 = (MddAddD3311 + MddAddD3312 + MddAddD3313 + MddAddD3321 + MddAddD3322 + MddAddD3323
+ MddAddD3331 + MddAddD3332 + MddAddD3333);

%% CALCULATION FOR I = 3

% Calculation of the 1,1 Component of the LddD 2nd Order Tensor

MddAddD1111 = eigAvec3(1)^4                          *A11IBOF_val;
MddAddD1112 = eigAvec3(1)^3 *eigAvec3(2)             *A12IBOF_val;
MddAddD1113 = eigAvec3(1)^3              *eigAvec3(3)*A13IBOF_val;
MddAddD1121 = eigAvec3(1)^3 *eigAvec3(2)             *A12IBOF_val;
MddAddD1122 = eigAvec3(1)^2 *eigAvec3(2)^2           *A22IBOF_val;
MddAddD1123 = eigAvec3(1)^2 *eigAvec3(2)  *eigAvec3(3)*A23IBOF_val;
MddAddD1131 = eigAvec3(1)^3              *eigAvec3(3)*A13IBOF_val;
MddAddD1132 = eigAvec3(1)^2 *eigAvec3(2)  *eigAvec3(3)*A23IBOF_val;
MddAddD1133 = eigAvec3(1)^2              *eigAvec3(3)^2*A33IBOF_val;

MddAddD11_e3 = (MddAddD1111 + MddAddD1112 + MddAddD1113 + MddAddD1121 + MddAddD1122 + MddAddD1123
+ MddAddD1131 + MddAddD1132 + MddAddD1133);

% Calculation of the 1,2 Component of the MddAddD 2nd Order Tensor
```

```
MddAddD1211 = eigAvec3(1)^3 *eigAvec3(2)                *A11IBOF_val;
MddAddD1212 = eigAvec3(1)^2 *eigAvec3(2)^2              *A12IBOF_val;
MddAddD1213 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *A13IBOF_val;
MddAddD1221 = eigAvec3(1)^2 *eigAvec3(2)^2              *A12IBOF_val;
MddAddD1222 = eigAvec3(1)   *eigAvec3(2)^3              *A22IBOF_val;
MddAddD1223 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *A23IBOF_val;
MddAddD1231 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *A13IBOF_val;
MddAddD1232 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *A23IBOF_val;
MddAddD1233 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2  *A33IBOF_val;

MddAddD12_e3 = (MddAddD1211 + MddAddD1212 + MddAddD1213 + MddAddD1221 + MddAddD1222 + MddAddD1223
+ MddAddD1231 + MddAddD1232 + MddAddD1233);

% Calculation of the 1,3 Component of the MddAddD 2nd Order Tensor

MddAddD1311 = eigAvec3(1)^3             *eigAvec3(3)    *A11IBOF_val;
MddAddD1312 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *A12IBOF_val;
MddAddD1313 = eigAvec3(1)^2             *eigAvec3(3)^2   *A13IBOF_val;
MddAddD1321 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *A12IBOF_val;
MddAddD1322 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *A22IBOF_val;
MddAddD1323 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *A23IBOF_val;
MddAddD1331 = eigAvec3(1)^2             *eigAvec3(3)^2   *A13IBOF_val;
MddAddD1332 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *A23IBOF_val;
MddAddD1333 = eigAvec3(1)               *eigAvec3(3)^3   *A33IBOF_val;

MddAddD13_e3 = (MddAddD1311 + MddAddD1312 + MddAddD1313 + MddAddD1321 + MddAddD1322 + MddAddD1323
+ MddAddD1331 + MddAddD1332 + MddAddD1333);

% Calculation of the 2,1 Component of the MddAddD 2nd Order Tensor

MddAddD21_e3 = MddAddD12_e3;    %DUE TO SYMMETRY!

% Calculation of the 2,2 Component of the MddAddD 2nd Order Tensor

MddAddD2211 = eigAvec3(1)^2 *eigAvec3(2)^2              *A11IBOF_val;
MddAddD2212 = eigAvec3(1)   *eigAvec3(2)^3              *A12IBOF_val;
MddAddD2213 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *A13IBOF_val;
MddAddD2221 = eigAvec3(1)   *eigAvec3(2)^3              *A12IBOF_val;
MddAddD2222 =               eigAvec3(2)^4               *A22IBOF_val;
MddAddD2223 =               eigAvec3(2)^3 *eigAvec3(3)    *A23IBOF_val;
MddAddD2231 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *A13IBOF_val;
MddAddD2232 =               eigAvec3(2)^3 *eigAvec3(3)    *A23IBOF_val;
MddAddD2233 =               eigAvec3(2)^2 *eigAvec3(3)^2   *A33IBOF_val;

MddAddD22_e3 = (MddAddD2211 + MddAddD2212 + MddAddD2213 + MddAddD2221 + MddAddD2222 + MddAddD2223
+ MddAddD2231 + MddAddD2232 + MddAddD2233);

% Calculation of the 2,3 Component of the MddAddD 2nd Order Tensor

MddAddD2311 = eigAvec3(1)^2 *eigAvec3(2)   *eigAvec3(3)    *A11IBOF_val;
MddAddD2312 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *A12IBOF_val;
MddAddD2313 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *A13IBOF_val;
MddAddD2321 = eigAvec3(1)   *eigAvec3(2)^2 *eigAvec3(3)    *A12IBOF_val;
MddAddD2322 =               eigAvec3(2)^3 *eigAvec3(3)    *A22IBOF_val;
MddAddD2323 =               eigAvec3(2)^2 *eigAvec3(3)^2   *A23IBOF_val;
MddAddD2331 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *A13IBOF_val;
MddAddD2332 =               eigAvec3(2)^2 *eigAvec3(3)^2   *A23IBOF_val;
```

```
MddAddD2333 =              eigAvec3(2)   *eigAvec3(3)^3   *A33IBOF_val;

MddAddD23_e3 = (MddAddD2311 + MddAddD2312 + MddAddD2313 + MddAddD2321 + MddAddD2322 + MddAddD2323
+ MddAddD2331 + MddAddD2332 + MddAddD2333);

% Calculation of the 3,1 Component of the MddAddD 2nd Order Tensor

MddAddD31_e3 = MddAddD13_e3;    %DUE TO SYMMETRY!

% Calculation of the 3,2 Component of the MddAddD 2nd Order Tensor

MddAddD32_e3 = MddAddD23_e3;    %DUE TO SYMMETRY!

% Calculation of the 3,3 Component of the MddAddD 2nd Order Tensor

MddAddD3311 = eigAvec3(1)^2              *eigAvec3(3)^2   *A11IBOF_val;
MddAddD3312 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *A12IBOF_val;
MddAddD3313 = eigAvec3(1)                *eigAvec3(3)^3   *A13IBOF_val;
MddAddD3321 = eigAvec3(1)   *eigAvec3(2)   *eigAvec3(3)^2   *A12IBOF_val;
MddAddD3322 =              eigAvec3(2)^2 *eigAvec3(3)^2   *A22IBOF_val;
MddAddD3323 =              eigAvec3(2)   *eigAvec3(3)^3   *A23IBOF_val;
MddAddD3331 = eigAvec3(1)                *eigAvec3(3)^3   *A13IBOF_val;
MddAddD3332 =              eigAvec3(2)   *eigAvec3(3)^3   *A23IBOF_val;
MddAddD3333 =                          eigAvec3(3)^4   *A33IBOF_val;

MddAddD33_e3 = (MddAddD3311 + MddAddD3312 + MddAddD3313 + MddAddD3321 + MddAddD3322 + MddAddD3323
+ MddAddD3331 + MddAddD3332 + MddAddD3333);

%% Summing the terms over i (i.e. i = 1,2,3)

MddAddD11 = MddAddD11_e1 + MddAddD11_e2 + MddAddD11_e3;
MddAddD12 = MddAddD12_e1 + MddAddD12_e2 + MddAddD12_e3;
MddAddD13 = MddAddD13_e1 + MddAddD13_e2 + MddAddD13_e3;
MddAddD21 = MddAddD21_e1 + MddAddD21_e2 + MddAddD21_e3;
MddAddD22 = MddAddD22_e1 + MddAddD22_e2 + MddAddD22_e3;
MddAddD23 = MddAddD23_e1 + MddAddD23_e2 + MddAddD23_e3;
MddAddD31 = MddAddD31_e1 + MddAddD31_e2 + MddAddD31_e3;
MddAddD32 = MddAddD32_e1 + MddAddD32_e2 + MddAddD32_e3;
MddAddD33 = MddAddD33_e1 + MddAddD33_e2 + MddAddD33_e3;

%% Forming the M:A:D Tensor (2nd Order)

MddAddD = [MddAddD11 MddAddD12 MddAddD13 MddAddD21 MddAddD22 MddAddD23 MddAddD31 MddAddD32
MddAddD33];

MddAddD(isnan(MddAddD)) = 0;
MddAddD(isinf(MddAddD)) = 0;
```