

Development of a simulation tool to analyze the orientation of LCPs  
during extrusion process

by

Arash Ahmadzadegan

B.Sc., Shahrekord University, Iran, 2004

M.Sc., Amirkabir University of Technology, Iran, 2007

A dissertation submitted in partial fulfillment of the  
requirements for the Degree of Doctor of Philosophy  
in the Department of Mechanical Engineering at Tufts University

Medford, Massachusetts

August 2013

© Copyright 2013 by Arash Ahmadzadegan

Abstract of “Liquid Crystalline Polymers”, by Arash Ahmadzadegan, Ph.D., Tufts University, August 2013.

In this thesis, different aspects of the rheology and directionality of the liquid crystalline polymers (LCPs) are investigated. The rheology of LCPs are modeled with different rheological models in different die geometries. The final goal in modeling the rheology and directionality of LCPs is to have a better understanding of their rheology during extrusion processing methods inside extrusion dies and eventually produce more isotropic films of LCPs. An attempt to design a die geometry that produces more isotropic films was made and it was shown that it is possible to use the inertia of the polymer to generate a more isotropic velocity profile at the lip of the die. This isotropic velocity profile can lead to alignment of directors along the streamlines and produce an isotropic film of LCP. It is shown that the rheological properties of the LCP should be altered to have a very low viscosity for this type of die to work.

To be able to investigate the effect of processing on directionality of LCPs, it is essential to develop a method to simulate the directionality based on processing conditions. As a result, a user defined function (UDF) code was added to ANSYS® FLUENT® to simulate the directionality of LCPs. The rheology of the LCP is modeled using power-law fluid model and the consistency index (K) and power-law index (n) were estimated based on the experimental measurements done with capillary rheometry. Three main phenomena that affect the directionality namely effects of Franks elastic energy, the effect of shear and the effect of movement of crystals with the bulk of polymer are investigated. The results of this simulation are close to physical phenomena seen in real LCPs. To quantify the directionality of the LCPs, the order parameter of the domain were calculated and compared for different flow and fluid conditions.

All polymers including LCPs are viscoelastic fluids in molten state. To understand the rheology of LCPs, a die-swell experiment was carried out using LCP material and Polypropylene (PP). For this experiment a capillary die with two different land-lengths was designed and built. The die-swell of the materials were measured optically according to ISO standards and the dependence of the die swell for materials on rheological properties is investigated.

To simulate the viscoelasticity of LCPs numerically, ANSYS® POLYFLOW® was used. ANSYS® POLYFLOW® has several viscoelastic models and is designed to simulate

extrusion processes. The geometry of the capillary die designed for the experiments was modeled in ANSYS® POLYFLOW® and the results were compared with the experimental results obtained for LCP and PP. It is shown that the morphology of the polymer should be considered into account to have a correct simulation of die swell.

## **Dedication**

This thesis is dedicated to my wife and parents who helped me through the hard times and for their love and encouragements.

# Acknowledgements

First and foremost I offer my sincere gratitude to my advisors, Professor Michael Zimmerman and Professor Anil Saigal, who guided me during the majority of my years as a student at Tufts University. their advice, knowledge and encouragement, helped me achieve the level of my Ph.D degree, and evolve from the stage of a college student to a researcher.

Finally, my greatest thank goes to my friends and family, for standing by my side during every step of the way to the completion of this thesis. This thesis would not have been possible if it werent for the everyday support and encouragement of my parents. Although they were back home in Iran, they shared all my anxiety, troubles and successes, and made me feel they were with me all the way.

I also want to thank Roselita Fragoudakis, Professor Luisa Chiesa, Professor Richard Welzien, Professor Behrouz Abedian and Lorin Polidora for their help during the various stages of my PhD.

# Contents

<b>List of Tables</b>	<b>viii</b>
<b>List of Figures</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Modeling Rheology</b>	<b>6</b>
2.1 Cross-flow die . . . . .	6
2.1.1 Grid Generation . . . . .	9
2.1.2 Material Properties and boundary conditions . . . . .	10
2.1.3 Solution consideration and results . . . . .	11
2.2 Modeling the rheology and temperature distribution in a coat-hanger die . .	16
2.2.1 Modeling the coat-hanger die and the flow channel . . . . .	17
2.2.2 Modeling the flow channel of the coat-hanger die . . . . .	21
2.3 Rheological modeling on the structured and unstructured meshes for the directionality modeling . . . . .	34
2.4 Die Swell Measurements . . . . .	39
2.4.1 Numerical Simulation . . . . .	39
2.4.1.1 Phan Thien-Tanner Viscoelastic Model . . . . .	41
2.4.1.2 Rheological Parameters . . . . .	43
2.4.2 Experimental Measurements . . . . .	44
2.4.2.1 ISO Standard for Extrudate Swelling . . . . .	47
2.4.2.2 Design of the Extrusion Die . . . . .	48
2.4.2.3 Experimental Procedure . . . . .	53
2.4.3 Results . . . . .	60

<b>3</b>	<b>Modeling Directionality</b>	<b>61</b>
3.1	Introduction . . . . .	61
3.2	Modeling LCPs in macroscopic scale . . . . .	63
3.3	Initial and Boundary Conditions . . . . .	64
3.4	Franks Elastic Energy . . . . .	65
3.5	Evolution Equation . . . . .	67
3.6	Translation of Directors . . . . .	69
3.7	Order Parameter . . . . .	69
3.8	Numerical Method . . . . .	73
<b>4</b>	<b>Conclusion and Future Work</b>	<b>81</b>
<b>A</b>	<b>Introduction to UDFs</b>	<b>87</b>
<b>B</b>	<b>Non-dimensional numbers</b>	<b>88</b>
<b>C</b>	<b>Polymat’s material data for LCP (MAZ078MMR2.0)</b>	<b>91</b>
<b>D</b>	<b>UDF code for simulating the directionality</b>	<b>97</b>



# List of Tables

2.1	Constants used for the non-Newtonian power-law model . . . . .	12
2.2	Summary of the simulation results . . . . .	16
2.3	Thermal properties of the LCP as a function of temperature . . . . .	20
2.4	Activation energy/R for some polymers . . . . .	21
2.5	Temperature from the hopper to the lip of the die . . . . .	53
2.6	Die swell ratio for PP and LCP for different average velocities and land lengths	57

# List of Figures

1.1	General approach to find an extrusion die geometry with a more isotropic film	4
2.1	Geometry of the co-rotating biaxial die . . . . .	7
2.2	Geometry of a Coat-hanger die . . . . .	8
2.3	Geometry of a cross-flow die . . . . .	9
2.4	Structured mesh on the cross-flow geometry . . . . .	9
2.5	Unstructured grid on the geometry of the coat-hanger die . . . . .	10
2.6	Viscosity of LCP as a function of shear rate from a capillary rheometer . .	11
2.7	Velocity vectors across the thickness at the lip of the die (Newtonian Model)	13
2.8	Velocity vectors across the thickness at the lip of the die (Non-Newtonian Model) . . . . .	14
2.9	Change of angle between vectors as a function of the Re of the flow (Newtonian Model) . . . . .	15
2.10	Cross section of the Coat-hanger die . . . . .	18
2.11	Mesh structure inside the die and the flow channel . . . . .	19
2.12	Temperature contours on the surface of the coat-hanger die . . . . .	22
2.13	Simulated temperature contour on the surface of the flow channel . . . . .	22
2.14	Temperature contour at the horizontal cross section of the die and the flow channel . . . . .	23
2.15	Temperature contour at the vertical cross section of the die and the flow channel . . . . .	23
2.16	Velocity profile at the mid-planes of the die . . . . .	24
2.17	Velocity distribution at the lip of the die with isothermal boundary conditions	25
2.18	Measured Temperature Distribution . . . . .	25
2.19	List of measured points and their respective temperatures imported in Mathcad	27
2.20	Position of the imported point into Mathcad as seen normal to the xy-plane	28

2.21	Position of the imported point into Mathcad as seen from a 45 degrees (the vertical axis is the temperature) . . . . .	29
2.22	Second order regression of the temperature data in Mathcad . . . . .	30
2.23	Third order regression of the temperature data in Mathcad . . . . .	31
2.24	Parameters for the fourth order regression of the temperature data in Mathcad . . . . .	32
2.25	Surface plot of the fourth order regression of the temperature data in Mathcad . . . . .	33
2.26	Temperature boundary condition from the measured data . . . . .	35
2.27	Film extrusion showing higher velocity at the middle of the die . . . . .	36
2.28	Velocity profile at the lip of the die (from the center to the edge) . . . . .	36
2.29	Velocity Vectors at the lip of the die . . . . .	37
2.30	(a) Structured and (b) Unstructured meshes . . . . .	38
2.31	Shear viscosity vs. Shear rate for LCP at $T = 350^{\circ}C$ . . . . .	40
2.32	Generated mesh on the capillary die . . . . .	41
2.33	Change of elastic $G'$ and Loss $G''$ modulus with shear rate . . . . .	45
2.34	Die geometry (a) Wireframe (b) cross section . . . . .	46
2.35	Melt pipe connecting the extruder to the die. . . . .	46
2.36	DSM Xplore Micro 15cc twin screw compounder . . . . .	47
2.37	Temperature profile (a)with and (b) without heating elements . . . . .	50
2.38	5mm Capillary Die . . . . .	51
2.39	15mm Capillary die attachment . . . . .	52
2.40	Setup of the experiment . . . . .	54
2.41	Die swell for PP at different shear rates. . . . .	55
2.42	Die swell for LCP at different shear rates. . . . .	56
2.43	Comparison between die swell of LCP and PP for different land length and shear rate . . . . .	58
2.44	Velocity profile at the Lip of the Die . . . . .	59
3.1	Comparison between amorphous, semicrystalline and liquid crystalline polymers . . . . .	62
3.2	Orientation calculation flowchart . . . . .	65
3.3	Boundary conditions on directors . . . . .	65
3.4	Franks elastic energy minimization flowchart . . . . .	68
3.5	Translation of directors . . . . .	70
3.6	Effect of Frank elastic energy on directors in (a) structured mesh and (b) unstructured mesh . . . . .	74

3.7	Effect of shear on directors in (a) structured mesh and (b) unstructured mesh	75
3.8	Effect of translation of directors with bulk of fluid on structured mesh . . .	76
3.9	Effect of translation of directors with bulk of fluid on unstructured mesh . .	76
3.10	Final simulation of orientation (combination of all three steps) for (a) struc- tured mesh (b) unstructured mesh . . . . .	77
3.11	Effect of increasing the inlet velocity on the order parameter . . . . .	78
3.12	Effect of the power-law index on the order parameter . . . . .	79
3.13	Effect of the temperature difference between inlet and the walls on the order parameter . . . . .	79
3.14	Change of order parameter vs. Re when average velocity $\bar{V}$ is changing . .	80

# Chapter 1

## Introduction

With advances in technology and the introduction of new materials, the need for new processing methods to enable the benefits of these materials is growing. Liquid crystalline polymers (LCPs) are among a class of high performance polymers but the orientation of crystals in the final product can adversely affect their properties. Modeling the directionality of LCPs during manufacturing processes can provide information and insight into designing new processing methods in order to achieve desired material properties. Liquid crystalline polymers (LCPs) show some unique properties that make them suitable for special engineering applications. Among these properties are high mechanical strength at high temperatures, high chemical resistance, flame retardancy and good weatherability. They have also good electrical properties such as very low loss tangent and almost constant dielectric constant (Thompson et al. [1]). These exceptional properties even encouraged researchers to improve the properties of commercial polymers by using LCP as reinforcement (Chinsirikul et al. [2]). Most of the applications of LCPs are in injection molding processes. To utilize the exceptional properties of LCPs in extruded films, it is important to control the orientation of crystals in the film to achieve a more isotropic properties in the plane of the film. Different researchers have introduced different methods to make more isotropic films. Among these are the introduction of counter rotating dies by Lusignea [3], insertion of a porous media in the die by Boles et al. [4] and addition of a magnetic field by Fu and Wang [5]. During almost all manufacturing processes of LCPs, these materials are formed in the molten state. As a result, studying the rheology of LCPs is important and it is highly desirable to model their rheology computationally. Although amorphous polymers become isotropic in the molten state, LCPs have preferred orientation even in the molten state and the interaction between crystals plays an important role in rheology of

LCPs and causes some unique behaviors (Donald et al. [6]). Among these behaviors are low viscosity compared to conventional polymers and very small or no die swell during the extrusion. Moreover, because of the anisotropy of LCPs, the orientation of these crystals determines the properties of the final manufactured product. As an example, during the extrusion processes, crystals align themselves parallel to the direction of shear. But unlike amorphous polymers, molecules in LCPs have higher inertia which opposes the brownian motion of molecules and keeps their orientation even after exiting the die. This phenomenon makes the simulation of directionality inside the die more valuable as the directionality remains almost intact even after exiting the die. Although most of the theoretical research for modeling the rheology and orientation of LCPs is based on small molecule liquid crystals (SMLCs), the results of those studies are also applicable to polymeric liquid crystals, especially in the case of shear flows (Donald et al. [6]). The theories for the simulation of the rheology of LCPs are still evolving and researchers are trying to find more accurate theories (Han [7, 8]). The widely used constitutive equations relating the stress and strain for liquid crystals are based on the Leslie-Ericksen theory which was developed by Leslie [9] based on the work of Ericksen [10] and Frank [11]. These equations consider the effect of crystals on the stresses to be continuous and defines a vector field called the director field. The equations are very complicated and only simplified variations of them, namely transversely isotropic fluid (TIF) equations have been solved numerically on simple geometries (ex. Baleo et al. [12], Chang et al. [13]). These simplified equations have some restrictions as follows:

1. Flow domain is considered to be a mono-domain without disclinations
2. Due to the high viscosity approximation, elastic stresses are considered negligible compared to viscous stresses
3. The flow is considered steady and isothermal

These assumptions pose a significant restriction to the amount of experimental data available for verification since most of the experimental data on the rheology of LCPs are based on the observation of the disclination lines between polarized glasses. The other restriction of simulating the TIF equations numerically is that the convergence of these equations is very limited, especially in complex geometries. Since the geometries of the extrusion dies are normally complex and cannot always be discretized with a structured mesh, a practical and simplified method for simulating and approximating directionality on unstructured mesh is highly desirable. The method studied in this thesis has a Monte-Carlo basis and

can be applied to complex geometries. The other advantage of this method is the ability to be used on a combination of structured and unstructured meshes with actual fluid parameters without convergence concerns. As a result, it can be applied in practical situations. There are some major difficulties associated with modeling the directionality on unstructured meshes. The most important one is extending the director calculations to unstructured meshes. This problem is addressed here by treating each cell locally. This means that each cell interacts with only its neighbors. Although unstructured meshes introduce some errors associated with averaging of quantities, they are the only method of discretizing the complex geometries of dies.

The flowchart in Figure 1.1 shows the approach taken here to find an optimized geometry for making an isotropic film of LCP. In this approach, a geometry for the die is picked based on existing coat-hanger die design. In addition to the geometry, the rheological properties of the polymer are also important for the simulation and can be altered to change the rheology. After this step, the rheology of the polymer is modeled using a fluid model available based on the experimental measurements. This step involves meshing the geometry and applying the boundary conditions to the flow. The results of this simulation involves the velocity field, pressure field and temperature field. The velocity field is used in the next step to simulate the directionality of crystals. In this step a user defined function (UDF) is used to extract the velocity field from the rheological simulation and the constitutive equations for the Franks elastic energy, evolution equation and movement of crystals with the flow are applied to the crystals. In the next step after simulating the directionality it is important to quantify the directionality of crystals. This can be done using the order parameter. Order parameter is a quantity (for nematics is a scalar) that quantifies the degree of order in a LCP material. After quantifying the directionality it is possible to compare the directionality to a desired value and iterate the process by changing the geometry and/or rheological parameter to find a more isotropic LCP film.

Since the directionality modeling depends on the rheological simulation, it is important to be able to simulate the rheology of the polymer correctly. One of the important phenomena during the extrusion of plastics is the expansion of the extrudate after exiting the lip of the die. This phenomenon is commonly known as *extrudate swell* or *die swell*. There are two important reasons for the formation of the die swell.

First, the rapid changes in the velocity profile of the material leaving the inside of the die to the outside of the die. This difference in the velocity profile is the result of the polymer molecules attaching to the die causing zero velocity at the wall. The velocity of the polymer inside the die changes from zero at the wall to its maximum in the middle of

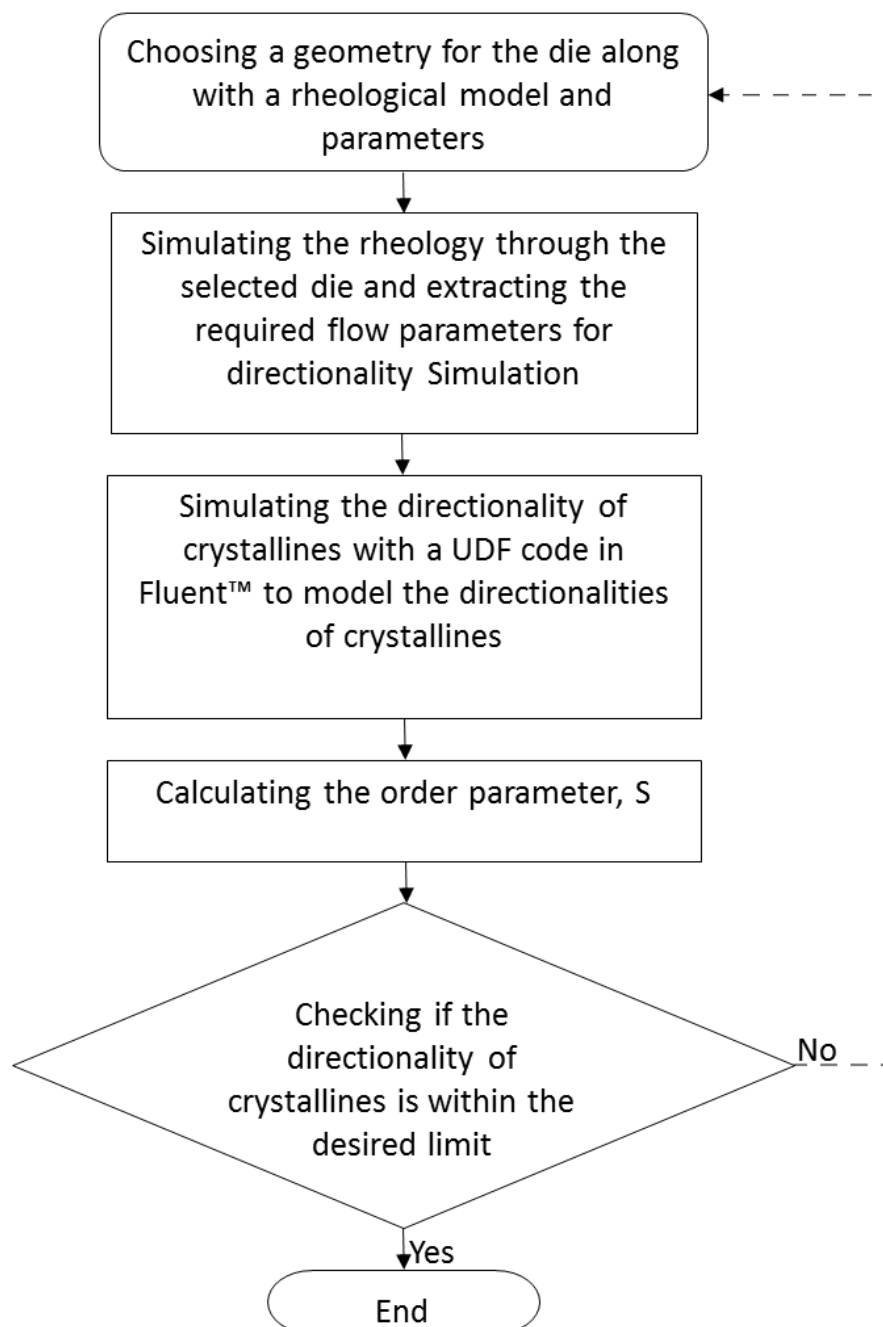


Figure 1.1: General approach to find an extrusion die geometry with a more isotropic film



the die. After leaving the die, extrudate moves with a constant velocity across its thickness. This means that the molecules closer to the wall need to accelerate and the molecules that were traveling at the center of the die need to decelerate and come to a mutual constant velocity. This causes the extrudate to swell as shown for Newtonian fluids by Hill and Chenier [14] experimentally and by Russo and Phillips [15] numerically.

Second reason is the elasticity of the polymers that is caused by the polymer molecules having a tendency to go to their previous state after the cessation of the shear. Due to the elasticity of the polymers, they show normal stress differences in presence of shear. These normal stresses are in addition to isotropic hydrostatic forces, do not exist in Newtonian fluids, and can be measured using various oscillatory rheometers [16, 17].

In this thesis, the die swell of a LCP material and a conventional polymer (PP) are compared experimentally. There is also an attempt to model the die swell of the LCP using Phan Thien-Tanner (PTT) viscoelastic model. The numerical simulation of the study is done using ANSYS<sup>®</sup> POLYFLOW<sup>®</sup>.

## Chapter 2

# Modeling Rheology

Modeling the rheology of the polymer is the first step in the simulation of the polymer behavior during the processing. Processing of the polymers include some complicated phenomena due to the viscoelastic behavior and thermal dependencies. For example, the temperature variations inside the die and flow channel is high enough to consider their effect on the viscosity of the polymer. In addition, due to the high viscosity of polymers, the heat generated due to the viscous dissipation should be taken into account. Another important characteristics of the polymers is the shear thinning behavior. This behavior can be modeled in ANSYS® FLUENT® using different fluid models. In this chapter, flow of polymers in different die geometries are simulated using ANSYS® FLUENT®. These geometries include a cross-flow die design, a coat-hanger design, a rectangular channel and a capillary die. The rectangular channel is used later to study the directionality of crystallines. The geometry for the capillary die is designed to measure and simulate the die swell of the different polymers. Die swell is one of the phenomena caused by the viscoelasticity of the polymers and gives some measures of the elasticity of the material. In this section, the die swell of polymers are measured experimentally and an attempt to numerically simulate the die swell with viscoelastic models of the ANSYS® POLYFLOW® is carried out.

### 2.1 Cross-flow die

Any rod like molecule or crystal tends to orient itself in the direction of shear as it moves within a flow. Since polymers consist of long chains of molecules, they experience the same phenomena when flowing in the melt condition. For most polymers, the molecules change direction and distribute uniformly in space during the solidification. However due to the higher inertia of the crystals in the liquid crystalline polymers, the oriented crystals will keep

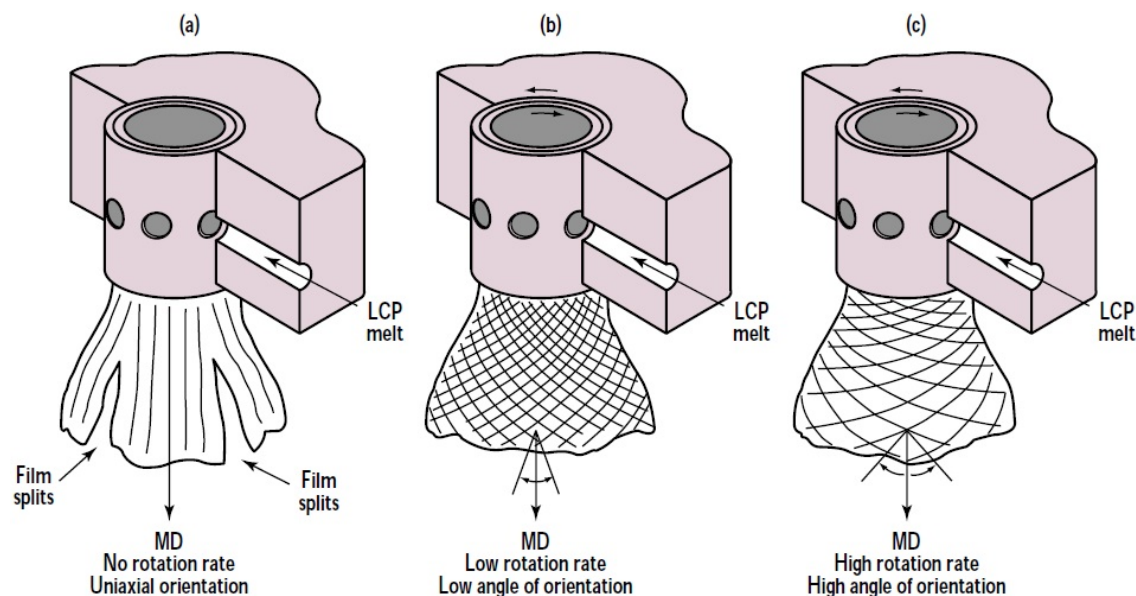


Figure 2.1: Geometry of the co-rotating biaxial die

their order during the solidification and as a result the final product will show anisotropy in the material properties. The product of interest here is the extruded film from the film extrusion process. To overcome this problem for extruded films, many complex die design technologies have been developed that involve moving surfaces. The use of biaxial shear flow during extrusion, elongational strain after extrusion, electromagnetic field effects, and thermal treatment to develop isotropic films has been discussed by Lusignea et al. [18]. Farrell and Lawrence use co-rotating extrusion dies to produce biaxially oriented films of LCPs [19]. The co-rotating extrusion dies try to use the viscosity of the polymer to produce a laminar flow with a lateral shear that has a three dimensional profile at the lip of the die. This geometry, as shown in Figure 2.1, has two co-rotating cylinder which operate at high temperature and the polymer melt flows between them. The change in the direction of the crystallines is introduced through the shear effect of the cylinders.

Due to the complexity of this type of die and built-in instabilities during this extrusion process, a simpler stationary extrusion die is more desirable.

An important parameter to be considered when designing the geometry of polymer dies is that the flow pattern inside the die highly influences the polymer extrusion process and any sharp edges inside the die which result in the flow to become turbulent should be avoided [20]. Turbulence inside the die increases the residual stresses which will be released after the extrudate leaves the die and causes instabilities in the extrusion process. The conventional die for extrusion of polymer films or the coat-hanger design is shown in Figure

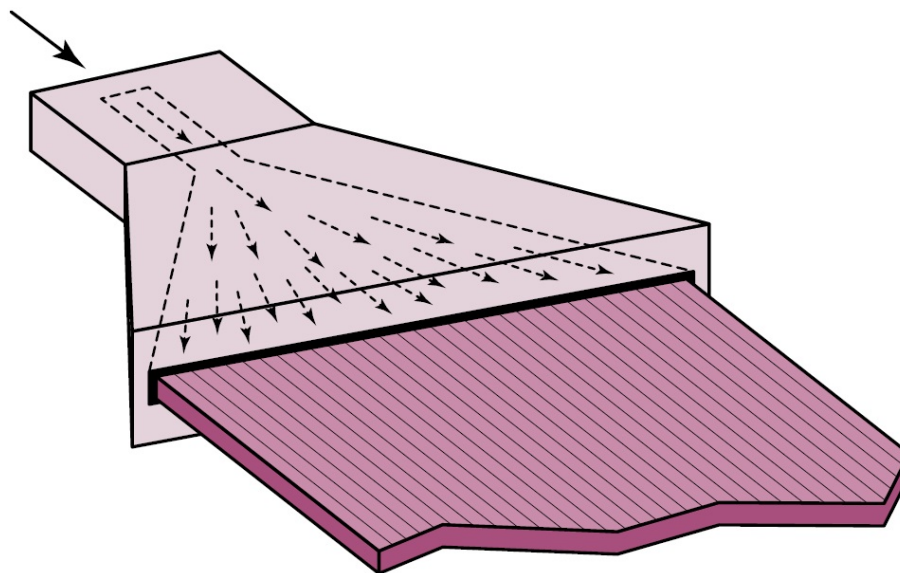


Figure 2.2: Geometry of a Coat-hanger die

2.2. These dies are designed such that the velocity of the extrudate coming out of the die is constant across the lip of the die.

A new design for the die geometry introduced here uses cross flows to produce the required lateral shear across the thickness of the film. As a result, no moving part is used in the die. A simple geometry of cross-flow die consists of two cross flows which interact with each other along the mid-plane. The interaction of these two symmetrically positioned flows causes the mid-plane to exit the lip normal to the exit plane while from the mid-plane to the upper and lower planes velocity vectors change direction gradually. To construct this geometry Gambit® was used since it can export the geometry, mesh and boundary conditions to FLUENT®. To demonstrate the idea of interacting cross flows, a model was built with 20 channels in which the average velocity vector of the flow in the top half of them is perpendicular to the average velocity vector in the bottom half. These channels are open along the interface between them and provide the interaction of flows inside the die (Figure 2.3).

This geometry can be constructed by machining the channels inside the upper and lower part of the die since the interface between them is open. It should also be noted that since the velocity of the polymer is inherently 3D in this flow, it is not possible to make a 2D geometry representing the cross flows.

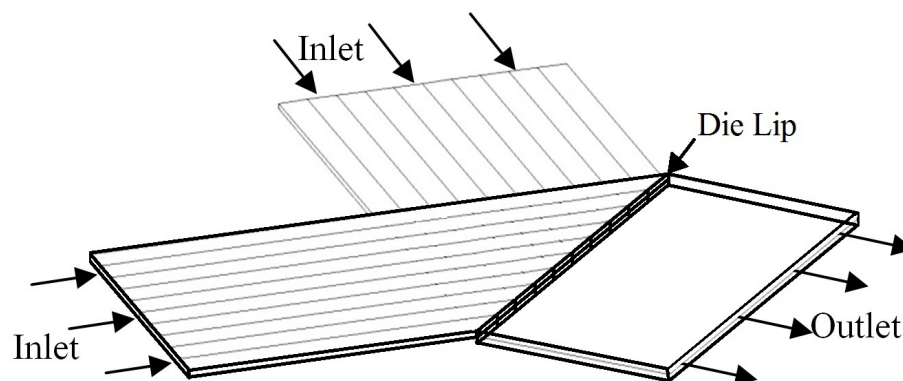


Figure 2.3: Geometry of a cross-flow die

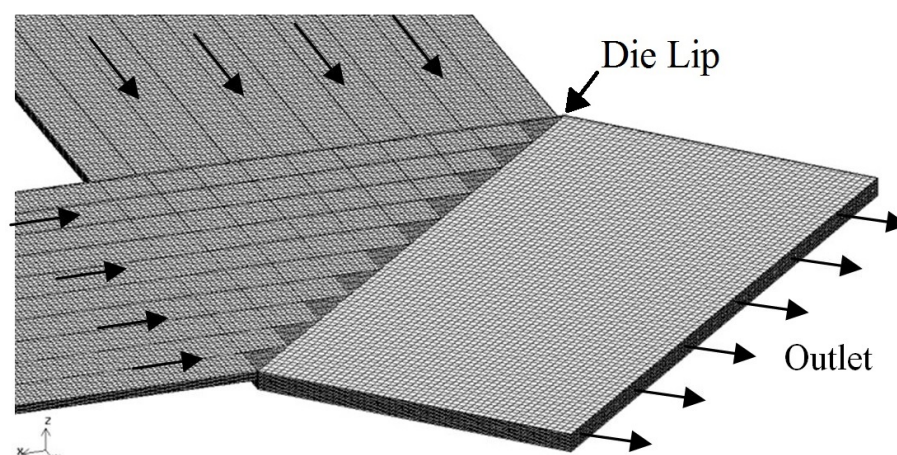


Figure 2.4: Structured mesh on the cross-flow geometry

### 2.1.1 Grid Generation

The required mesh to decompose the geometry to finite volumes was generated in GAMBIT®. Structured cubic elements were generated for the geometry of the cross-flow with the consideration of the flow pattern. Using hexahedral elements, it is possible to achieve higher accuracy in the simulation with fewer elements and make the simulation computationally cheaper. The maximum skewness of the generated grids is calculated to be 0.25 and the maximum aspect ratio of the elements is 7.1. Both these characteristics show that the mesh consists of high quality elements. The total number of elements inside and outside the die is 410,000. Figure 2.4 shows the generated mesh on the cross-flow geometry.

Due to the complexity of the coat-hanger die, unstructured hexahedral elements were chosen for decomposing the geometry into finite volumes. Figure 2.5 shows the mesh used for the coat-hanger geometry. Mesh density is reduced here for demonstration purposes.

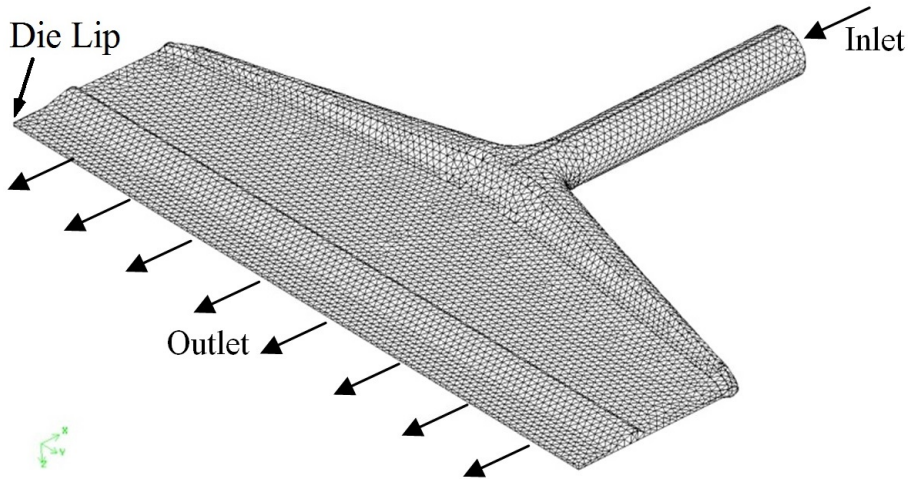


Figure 2.5: Unstructured grid on the geometry of the coat-hanger die

This mesh consists of 3,000,000 elements with the worst skewness on 0.777 and worst aspect ratio of 3.5 which are both in the range of acceptable limits.

### 2.1.2 Material Properties and boundary conditions

Two different models for material properties were used to simulate the flow inside the die. The first model considers the fluid to be Newtonian and defines a constant viscosity. In this model stress and strain inside the flow are linearly proportional. The second model was the power-law fluid model which was chosen based on the available experimental results for one type of LCP. This experiment is done with a capillary rheometer using a capillary length of 20mm and diameter of 1mm on a LCP. This experiment was carried out under isothermal condition where the temperature was kept constant at  $350^{\circ}C$ . The results are shown in Figure 2.6.

By looking at the curve of viscosity vs. strain rate on a log-log scale, it appears that it can be modeled by the non-Newtonian power-law model. Based on this information the non-Newtonian power-law model was used as a second model to simulate the flow inside the die. In the power law model, equation 2.1 is used in the constitutive equation to model viscosity in FLUENT®[21].

$$\eta_{min} < \eta = k\dot{\gamma}^{n-1}e^{T_0/T} < \eta_{max} \quad (2.1)$$

To find the parameters of this equation, the automatic option of POLYMAT® software was used. Moreover, for defining this model completely, the minimum and maximum values of the viscosity should be given. These values were chosen to be the maximum and minimum

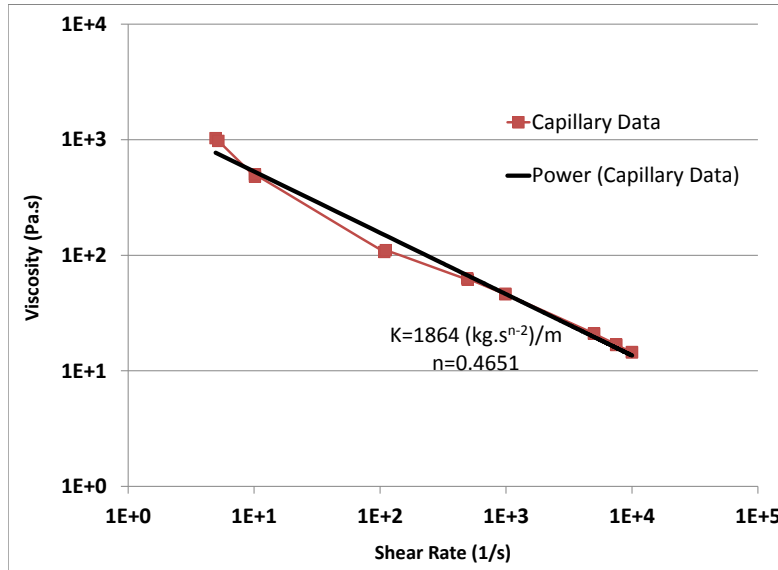


Figure 2.6: Viscosity of LCP as a function of shear rate from a capillary rheometer

values available from the capillary experiment which are  $1029 \text{ Kg}/(m.s)$  and  $14 \text{ Kg}/(m.s)$  respectively.

For finding a non-Newtonian model that satisfies the isotropy of extrudate, some different values for the non-Newtonian model parameters were considered. (Table 2.1)

Three different boundary conditions were used to define the boundaries of the geometry which include velocity inlet, walls and outlet. The fins inside the die were modeled as zero thickness walls with zero tangent and normal velocities. Also, since there is no free wall boundary condition available in FLUENT®<sup>®</sup>, the flow outside the die is also modeled inside a channel with slip condition (zero shear). It should be noted that the interfaces between the upper and the lower channels are defined as an interface. This means that two cross flows have interaction along this surface.

### 2.1.3 Solution consideration and results

FLUENT®<sup>®</sup> uses finite volume methods to solve the constitutive equations on the discretized flow field [21]. One important consideration before beginning the simulation is the importance of inertia terms in the equations. Since most polymers have high viscosities, flow is

Table 2.1: Constants used for the non-Newtonian power-law model

Consistency index (k) ( $Kg.s^{n-2}$ )/m	Power-Law index (n)	Max Viscosity Kg/(m.s)	Min Viscosity Kg/(m.s)	Velocity Vector Angle (degrees)
1864	0.4651	1029	14	0
1864	0.1	1029	0.0001	4
0.1	0.4651	1029	0.0001	0
0.1	0.2	1029	0.0001	15
0.01	0.4651	1029	0.0001	46
0.01	0.2	1029	0.0001	90 (desired)

mostly driven by pressure and the non-linear inertia terms can be neglected. The importance of inertia terms can be determined by calculating the Reynolds number of the flow. In all the simulations regarding the cross-flow, the inertia terms were considered. Ignoring the inertia terms helps with the convergence of the solution since these terms introduce non-linearity to the equations but it also reduces the accuracy of the results especially when the Re number is not small. Steady state, first order pressure based implicit solver was used in all cases.

The velocity vectors at the lip of the cross-flow die geometry for Newtonian fluid are shown in Figure 2.7. As can be seen, velocity vectors change from  $-45^\circ$  to  $45^\circ$  across the thickness of the film. Velocity vectors are always tangent to the streamlines.

Figure 2.8 shows the same geometry of cross-flow shown in Figure 2.7 with non-Newtonian power-law model. The parameters used for this simulation are from the capillary rheometer data shown in Figure 2.6. As can be seen, the change in the direction of velocity vectors across the die is very small and hard to recognize except for the vectors in the vicinity of the fins.

To study the effect of rheological model and properties on the velocity vectors at the lip of the die, the simulation of the rheology has been done using two different Newtonian and six non-Newtonian viscosity models. The first Newtonian model simulation on cross flow die geometry is done using a constant viscosity of 0.1 Pa.s. It can be seen from the velocity vectors that in this case the relative importance of inertia terms forces the fluid to keep its direction until it exits the die lip (Table 2.2). The second Newtonian model used the average value of the viscosities obtained during the capillary rheometry experiment which is 220.1 Pa.s. For this high value of viscosity, inertia forces are negligible compared to the viscous forces and flow reaches its fully developed condition in a very short distance and



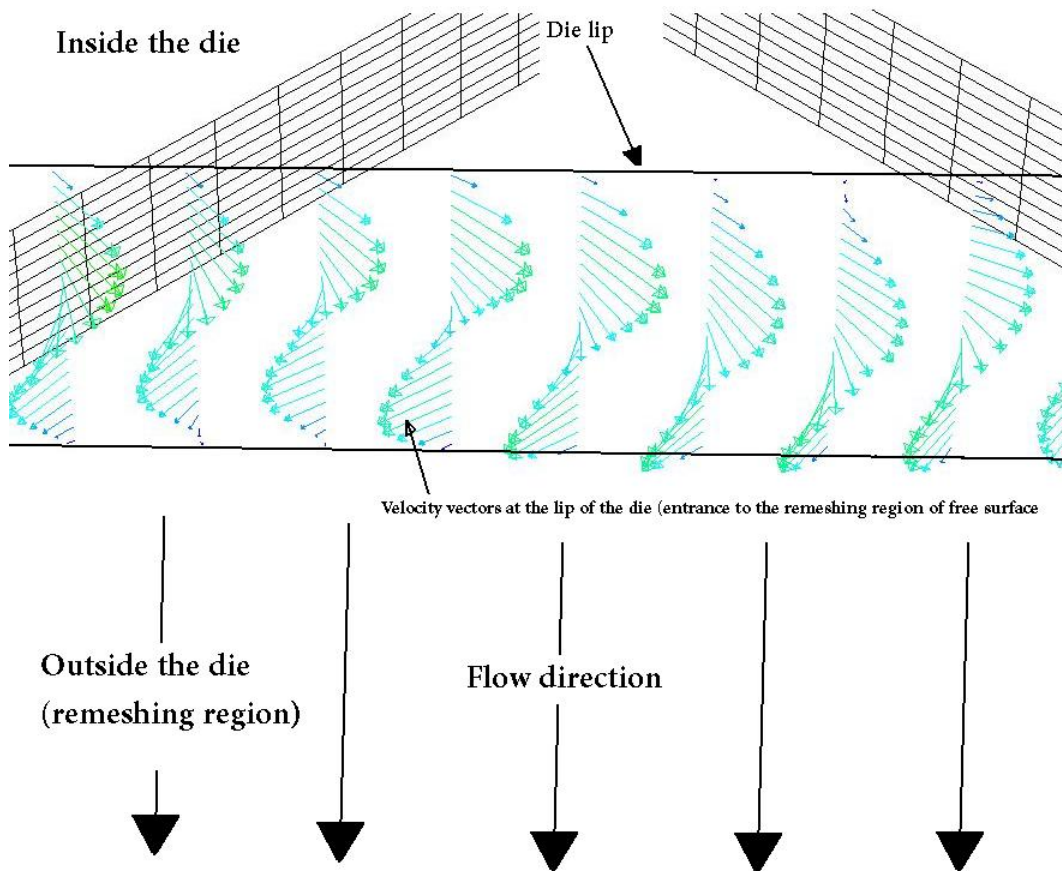


Figure 2.7: Velocity vectors across the thickness at the lip of the die (Newtonian Model)

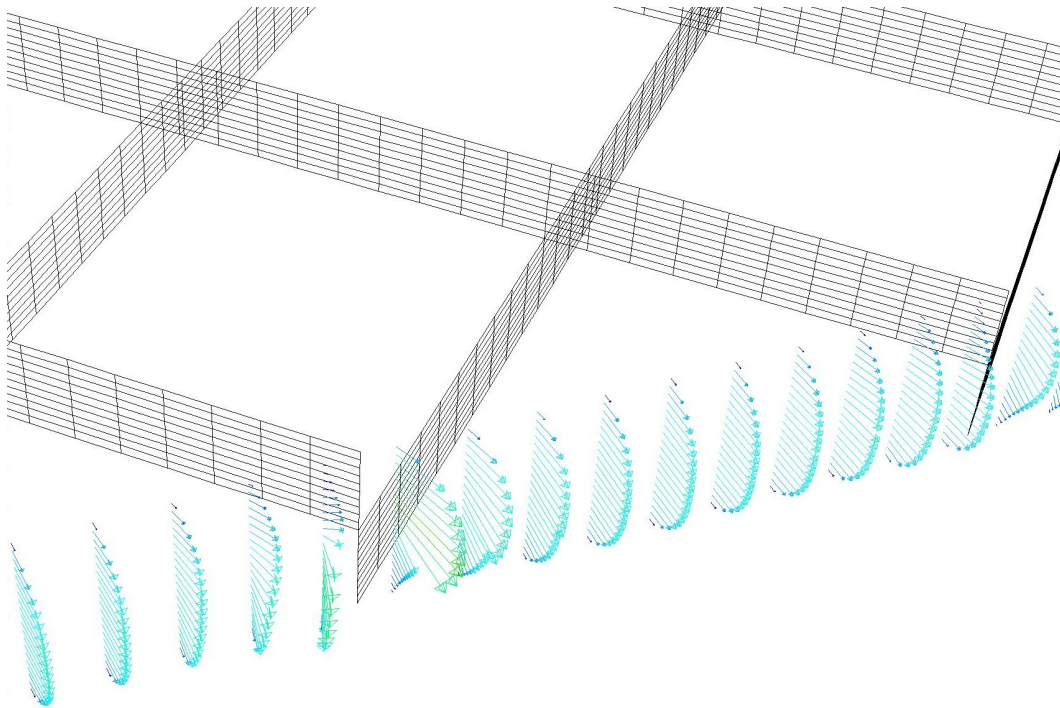


Figure 2.8: Velocity vectors across the thickness at the lip of the die (Non-Newtonian Model)

the only change in the velocity vectors are observed to be in vicinity of the fins. Although the actual material parameters for non-Newtonian model did not satisfy the isotropy in the plane of films, it is possible to change the material properties of the polymer to find a polymer that follows the power-law model and has the isotropy.

Since the maximum angle between two cross flows depends both on the inertia and viscous forces, this angle is derived for several different Reynolds numbers with Newtonian model. Re number is calculated with the characteristic length of the thickness of each cross flow channel. These results are shown in Figure 2.9.

By increasing the Re of the flow, the relative importance of the inertia forces increases and we approach the 90 degrees difference (from -45 to +45 degrees). On the other hand, at very low Re, flow approaches its fully developed condition and there will be no difference in the angle. According to Figure 2.9, it can be seen that there is a turning point at around  $Re=500$  that the angle is 80 degrees and at lower Re the angle does not satisfy the desired angle between the two flows.

The last simulation on the cross-flow geometry is done using the actual experimentally obtained rheological data. In this case, due to the shear thinning effect of the viscosity, the pressure drop is less than the pressure drop obtained for constant viscosity and also the

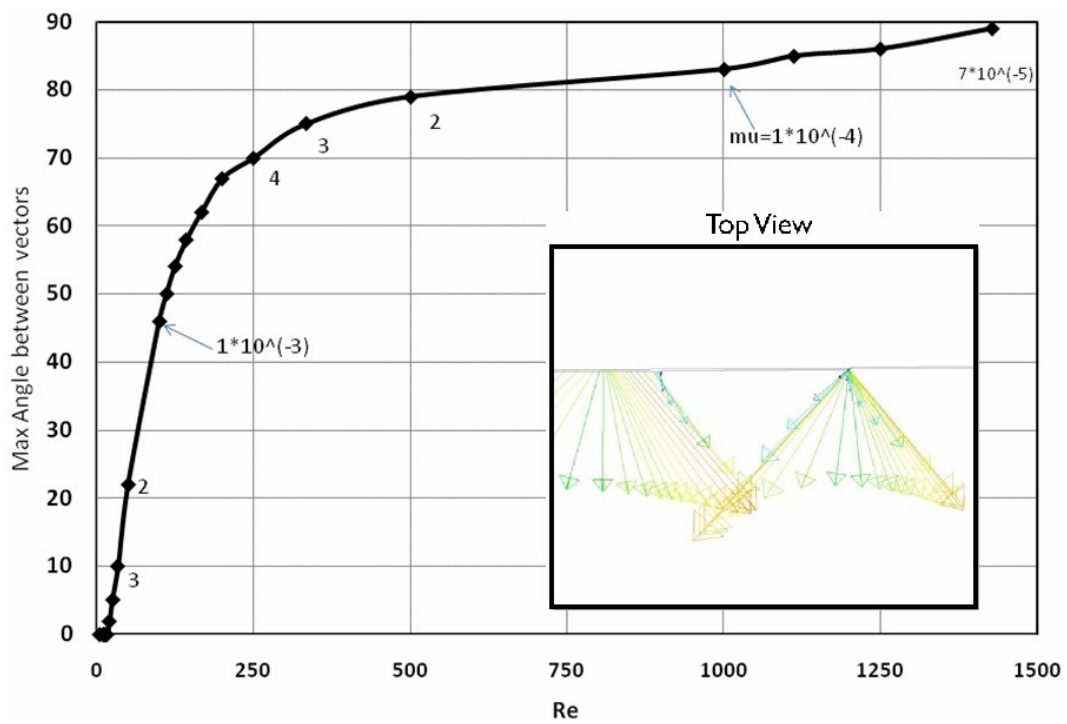
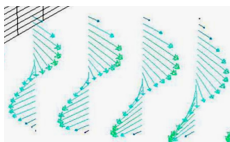
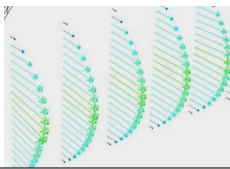
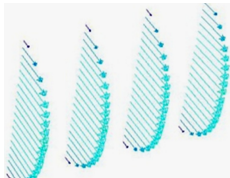


Figure 2.9: Change of angle between vectors as a function of the Re of the flow (Newtonian Model)

Table 2.2: Summary of the simulation results

.	$\Delta P(Pa)$	Velocity Profile
Newtonian coat-hanger ( $\eta = 220.1 Pa.s$ )	258,720	Aligned
Power-law coat-hanger ( $\eta = 1864\dot{\gamma}^{0.4651} Pa.s$ )	709,400	Aligned
Newtonian cross-flow with low viscosity ( $\eta = 0.1 Pa.s$ )	2476	
Newtonian cross-flow with high viscosity ( $\eta = 220.1 Pa.s$ )	4,207,000	
Power-law cross flow ( $\eta = 1864\dot{\gamma}^{0.4651} Pa.s$ )	3,224,000	

velocity profile at the lip is more flat.

Table 2.2 summarizes the results obtained from FLUENT® for the coat-hanger and the cross-flow geometries. From these results it can be observed that it is possible to obtain the correct molecular orientation using this new cross-flow die geometry. One solution for increasing the change in the direction of velocity vectors can be increasing the number of fins inside the die or increasing the angle between the two cross flows. It is also clear that the design of the die geometry is directly related to the material and processing characteristics such as viscosity and inlet velocities.

## 2.2 Modeling the rheology and temperature distribution in a coat-hanger die

To better understand the effect of temperature distribution on the rheology and the die, a simulation of the rheology with the consideration of temperature is done here. The design of a coat-hanger die is normally done considering an isothermal condition. This means

that for the die to operate at its design point, it is necessary to keep the whole die at a constant temperature. Any deviation from the isothermal condition causes the die not to operate at its design point. Coat-hanger dies are designed to provide a uniform velocity profile at the exit of the die. These type of dies are a way of distributing the polymer coming from the extruder in a cylindrical pipe and forming it in the shape of a thin film. To keep the die in isothermal condition, heating elements and thermocouples are used and a controller controls the power going into the heating elements based on the temperature data from the thermocouples. Figure 2.10 shows the coat-hanger die used in this study and the placements of the heating elements in it. Since the flow and geometry of the coat-hanger die has a symmetry plane in the middle, all the simulations are done using half of the die with a symmetry boundary condition. This approach reduces the number of cells to half which makes it possible to have a more accurate simulation in addition to being computationally cheaper.

Two different types of simulations was done on the coat-hanger die based on the type of boundary conditions in hand.

- In one case, thermal boundary conditions are defined on the body of the die. These boundary conditions are consist of the convection heat transfer to the ambient from the outer surface of the die and constant temperature at the position of the heating elements. In this case the temperature of the surface of the flow channel and the exchange of heat between the die and the polymer will be simulated as a part of the solution.
- The second type of simulation is done using the measured temperatures on the surface of the flow channel. In this case, the temperature of the surface is imposed and its effect on the rheology is simulated.

### **2.2.1 Modeling the coat-hanger die and the flow channel**

The importance of modeling the coat-hanger die in addition to the flow channel is that more realistic thermal boundary conditions can be applied used for the simulation. In the die, the only physical phenomena to be simulated is the conduction heat transfer using the Fourier's law. On the other hand, inside the flow channel Navier-Stokes equation should be solved in addition to the energy equation and the dependence of the solution on the mesh is more important. Based on this discussion, unstructured tetrahedral meshes are considered to discretize the geometry of the die and finer mostly structured meshes were used to model the rheology inside the flow channel. Figure 2.11 shows the symmetry plane

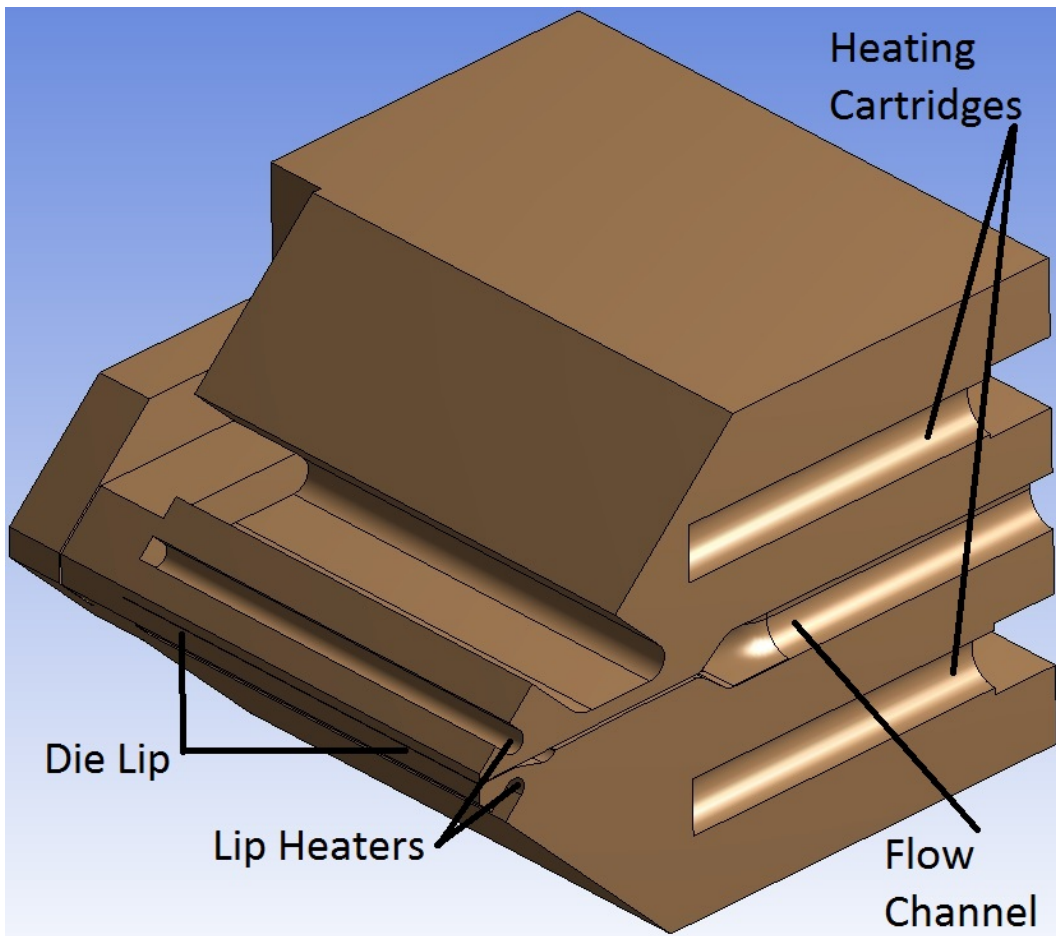


Figure 2.10: Cross section of the Coat-hanger die

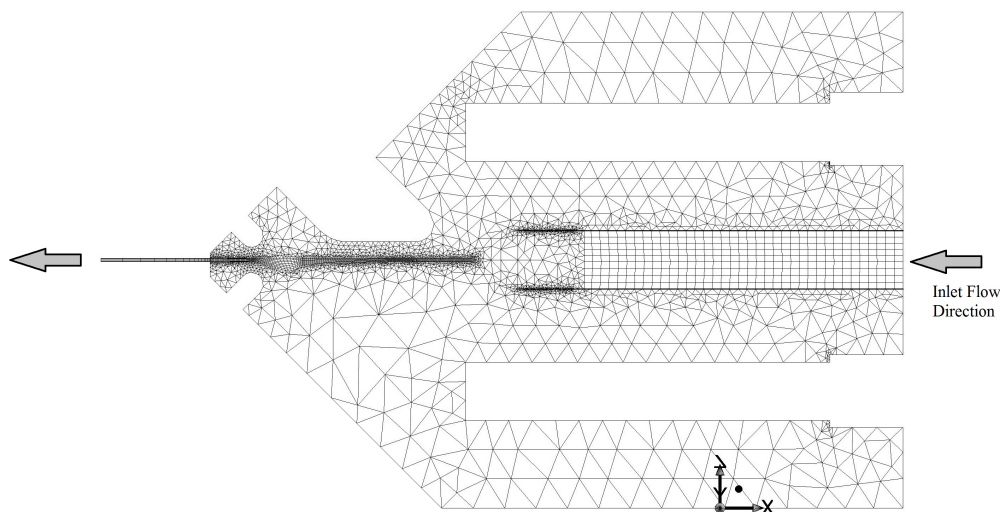


Figure 2.11: Mesh structure inside the die and the flow channel

of the die and flow channel. Due to the symmetry only half of the die needs to be simulated. As can be seen from Figure 2.11, the geometry of the flow channel is extended outside the die. This geometry is to apply the physical boundary conditions more realistically. The top and bottom surfaces of the extended flow channel have slip condition with zero shear. This simulates the free surface condition outside the die although the die swell and remeshing of the flow channel is not considered. The pressure is imposed at the end of this extended geometry not to disturb the simulated flow inside the flow channel. The thermal boundary condition on the extended geometry is considered to be convective heat transfer to the surroundings as it would be for a free surface of the polymer.

Other boundary conditions on the surfaces of the geometry include:

- **Convection heat transfer from the surface of the die.** The orientation and placements of the surfaces should be taken into account when defining the convection heat transfer coefficient. For example for the thin gaps between different parts of the die geometry the convection heat transfer coefficient of  $10 \text{ W}/(\text{m}^2 \cdot \text{K})$  and for the rest of the exposed surfaces the convection coefficient of  $100 \text{ W}/(\text{m}^2 \cdot \text{K})$  was defined.
- **Constant temperature at the heaters.** Although the heaters generate heat rather than having a constant temperature, the constant temperature boundary condition is defined for them with the set temperature for their respective thermocouple. It is possible to calculate the amount of heat generated from each of the heaters with this

Table 2.3: Thermal properties of the LCP as a function of temperature

Temperature (C)	Specific heat $C_p$ ( $J/(Kg.K)$ )		Temperature (C)	Thermal conductivity $k$ ( $W/(m.K)$ )
60	1055		68	0.367
90	1207		109	0.387
120	1310		129	0.37
150	1399		148	0.391
180	1473		168	0.385
210	1536		187	0.384
240	1605		207	0.374
270	1691		228	0.385
295	1770		248	0.371
320	1796		310	0.383
345	1732		330	0.349
370	1728		371	0.367

method and design the heaters based on this information.

- **Mass flow inlet.** The weight of the polymer coming out of the die lip was measured several times and the average value of these measurements was used as a mass flow inlet. It is assumed that the density of the polymer is the same in the molten and solid states.
- **No slip wall boundary condition.** The rheological boundary condition on the walls of the flow channel are wall boundary condition with no slip. The thermal boundary condition on this surface is the interface and has to be simulated as a part of the solution.

### Material properties

The die is made out of steel and its properties were extracted from the FLUENT® material library. These properties are specific heat of  $C_p = 502.48 J/(Kg.K)$  and thermal conductivity of  $k = 16.27 W/(m.K)$ . For the LCP these thermal properties can be entered as a function of temperature. These properties are measured and are given in terms of temperature in Table 2.3.

The effect of temperature on the viscosity of the LCP can be modeled using the Arrhenius model. This model is available in FLUENT® through equations 2.2 and 2.3.



Table 2.4: Activation energy/R for some polymers

Fluid	$\eta_0$ (Pa.s)	$T_0$ (K)	$E/R$ (K)
High-impact PS	$1.45 \times 10^5$	463	3707
PS	$9.2 \times 10^3$	483	4954
PP	$3.2 \times 10^3$	463	$5.1 - 5.6 \times 10^3$
HDPE	1520	473	$2.8 - 3.3 \times 10^3$
LDPE	3200	453	6840
Polymethyl methacrylate	6000	513	9855

$$\mu = \eta(\dot{\gamma}) \cdot H(T) \quad (2.2)$$

$$H(T) = \exp \left[ \alpha \left( \frac{1}{T - T_0} - \frac{1}{T_\alpha - T_0} \right) \right] \quad (2.3)$$

where  $\alpha$  is the ratio of the activation energy to the thermodynamics constant and  $T_\alpha$  is the reference temperature for which  $H(T) = 1$ .  $T_0$ , which is the temperature shift, is set to 0 by default and corresponds to the lowest temperature thermodynamically possible.  $T_0$  and  $T_\alpha$  are absolute temperatures.  $\alpha$  for some polymers are mentioned in Table 2.4.

## Results

Figures 2.12, 2.13, 2.14, 2.15 show the simulated temperature on the different parts of the die. It can be seen that the coolest part of the die is the end plate close to the end of the flow channel.

Since the effect of the temperature on the viscosity of the polymer is modeled here, it is possible to compare the flow of polymer with and without the effect of temperature variation on the viscosity. The best way to look at the effect of temperature on the rheology is to look at the velocity profile at the lip of the die. Figure 2.16 shows the velocity contours inside the flow channel on two perpendicular surfaces. It can be seen from Figure 2.12 that the lack of temperature control close to the lip of the die on both sides caused very low temperature compared to the ideal  $350^\circ$ . This lower temperature on the surface of the die causes the polymer inside the die to have lower temperature (Figure 2.14).

### 2.2.2 Modeling the flow channel of the coat-hanger die

The second type of simulation done for the rheology of the LCP uses the isothermal and measured temperature profiles inside the die. The measurements are done using thermocouples in the absence of the flow of polymer. Figure 2.17 shows the velocity distribution

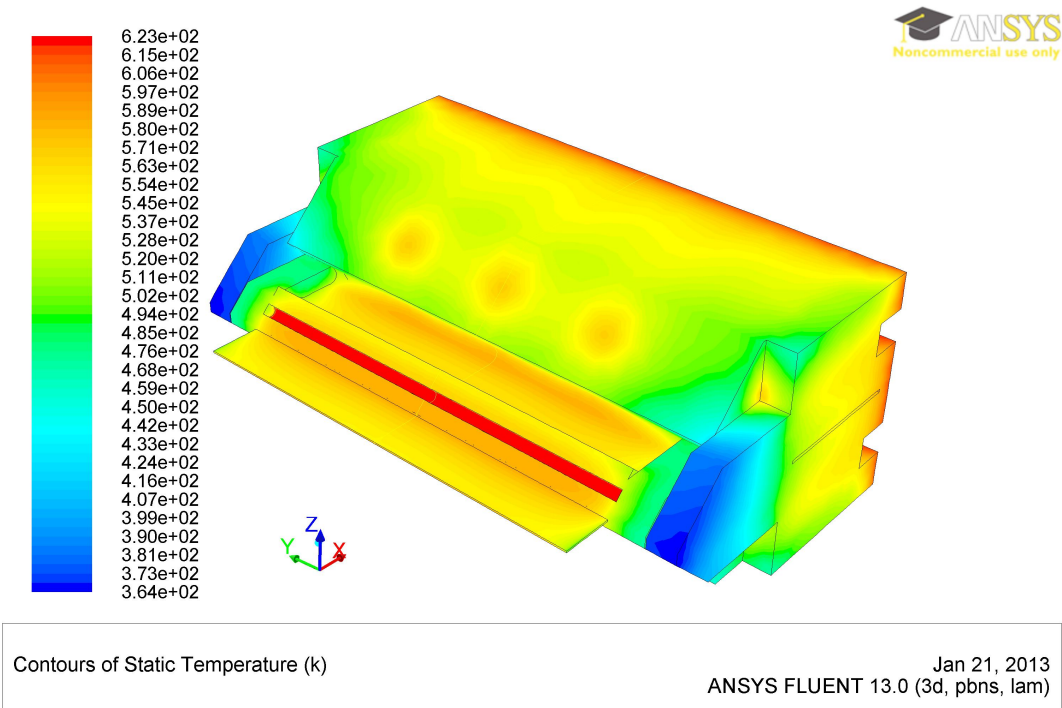


Figure 2.12: Temperature contours on the surface of the coat-hanger die

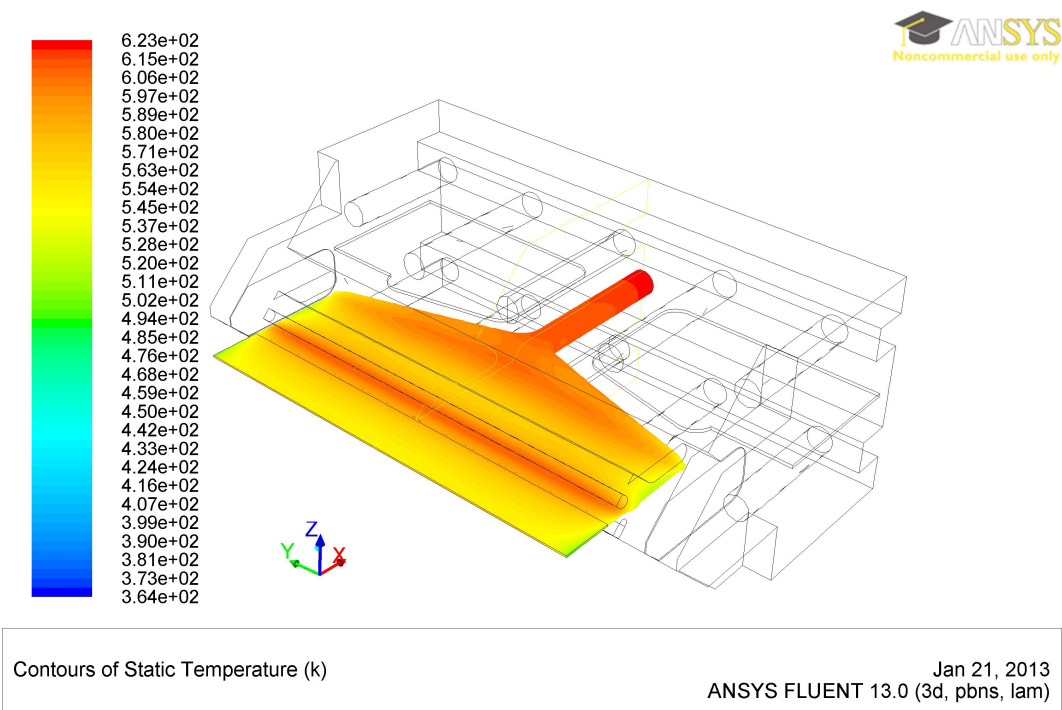


Figure 2.13: Simulated temperature contour on the surface of the flow channel

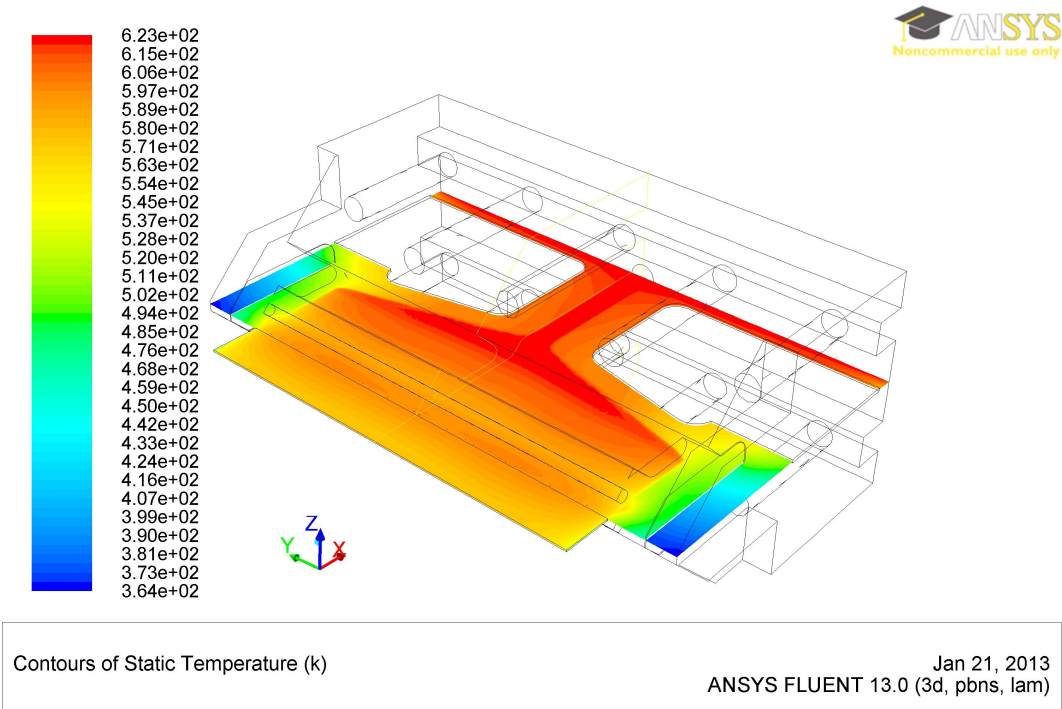


Figure 2.14: Temperature contour at the horizontal cross section of the die and the flow channel

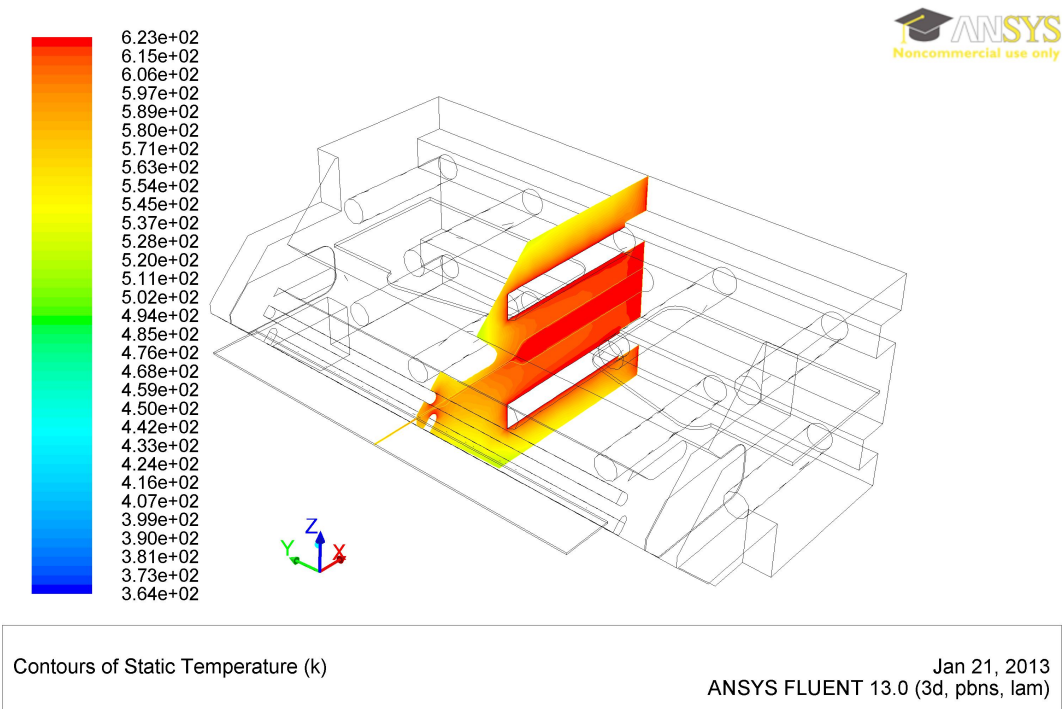


Figure 2.15: Temperature contour at the vertical cross section of the die and the flow channel

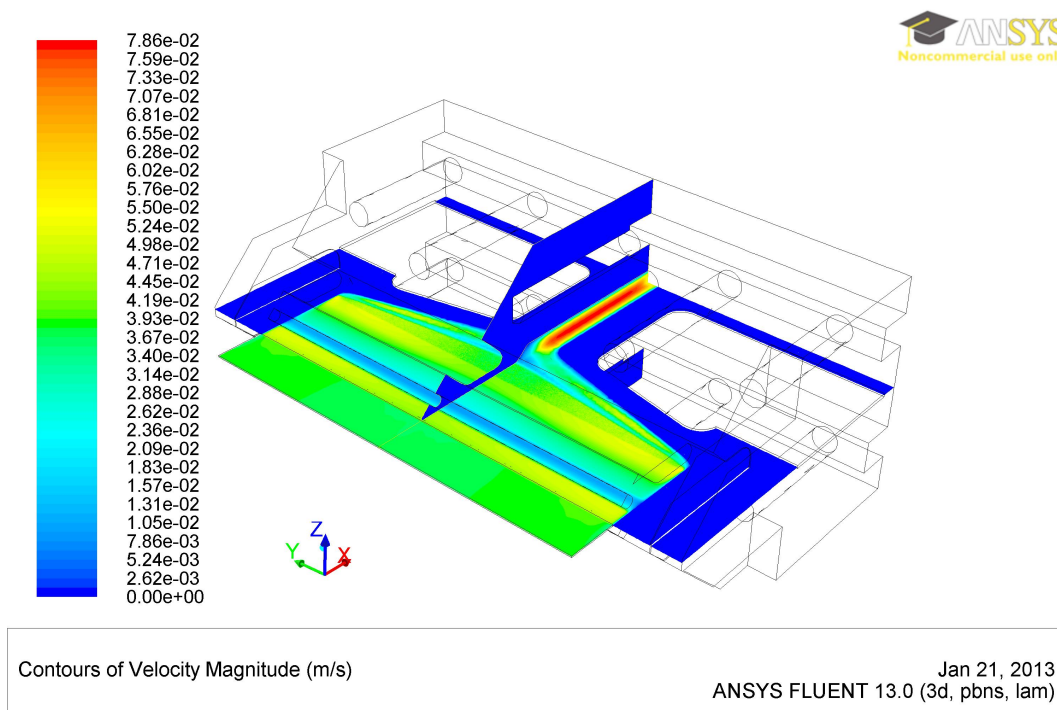


Figure 2.16: Velocity profile at the mid-planes of the die

at the lip of the die with the isothermal boundary conditions. As can be seen from Figure 2.17, the velocity profile has a very small variation across the lip of the die and has higher velocities closer to the corners of the die. This velocity distribution has not been observed in real extrusion processes. In real extrusion processes keeping the die in an isothermal condition is difficult. Figure 2.18 shows the actual measured which shows almost  $20^{\circ}\text{C}$  difference between maximum and minimum temperatures inside the die.

The measurements of temperature inside the die was done by attaching thermocouples to the surface of the flow channel in several positions as shown in Figure 2.18. These measurements are done along the lines that are supposed to be the same temperature. The goal here is to find the effect of the actual temperature profile on the rheology of the polymer inside the flow channel.

To import the measured temperature data to FLUENT® and use them as thermal boundary condition for the flow channel, the position of the data measured needs to match the geometry to be modeled. By knowing the scale of the flow channel, ImageJ image processing software was used to find the position of each of the points on the flow channel. Figure 2.19 shows the list of coordinates and their respective temperatures imported into Mathcad®. Mathcad® is going to be used to find a function for the temperature

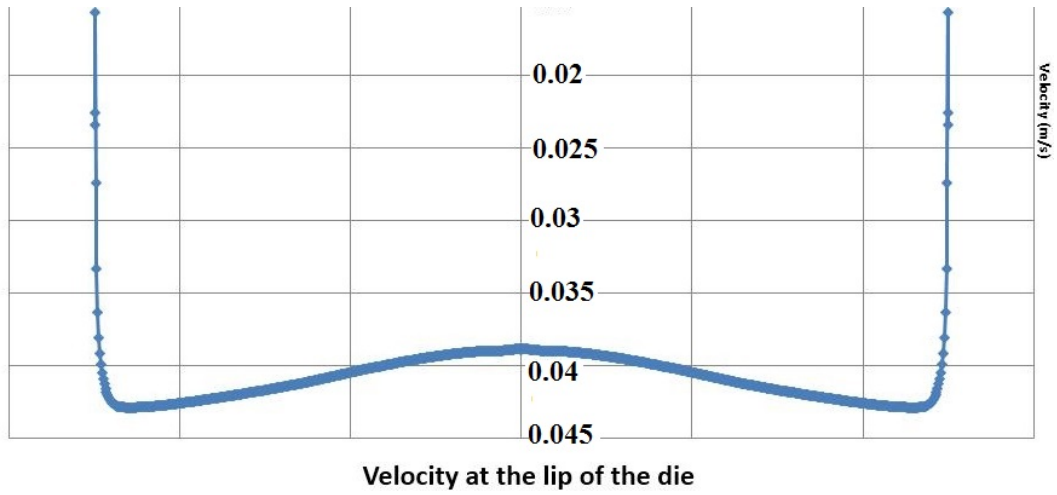


Figure 2.17: Velocity distribution at the lip of the die with isothermal boundary conditions

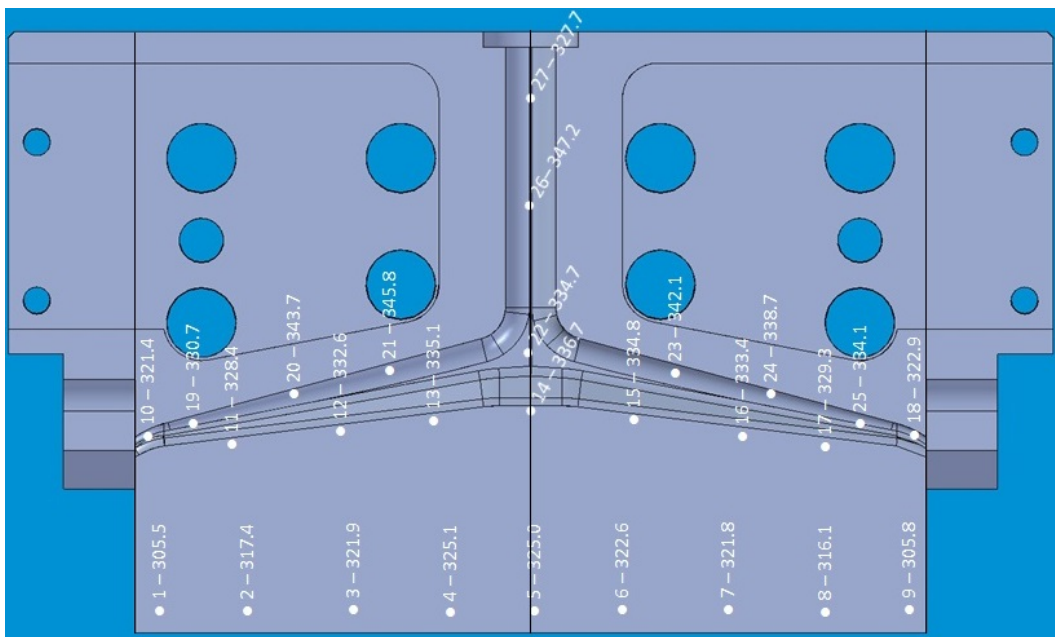


Figure 2.18: Measured Temperature Distribution

distribution.

Figure 2.20 shows the plotted points on the XY plane to verify the extracted data from ImageJ. It can be seen that if we look perpendicular to the XY axis the points are positioned correctly as measured.

A 45° view of the point with the z-axis indicating the temperature of each point is shown in Figure 2.21. As can be seen the temperature inside the flow channel has a maximum in the pipe that connects to the coat hanger and the temperature distribution is almost symmetrical with respect to the symmetry plane of the flow channel. This symmetry lets the simulation of the rheology to be done in half of the channel.

It is possible to apply a variable temperature as a boundary condition in FLUENT®. To do so we need to have the temperature as a function of coordinate system. To find this function, three different regressions were tried on the measured temperatures. The constants and plot of a second order regression for the temperature data are shown in Figure 2.22. A third order and a fourth order regressions are also tried and shown in Figures 2.23, 2.24 and 2.25. Since the temperature data is almost symmetric, the third order regression cannot follow the temperature trend. Between the second order and fourth order regressions, second order regression was chosen to apply the thermal boundary conditions with. This choice is due to the existence of the non physical maximum and minimum temperatures that can be seen in Figure 2.25.

The second order regression has been implemented as a thermal boundary condition with the following user defined function:

```
DEFINE_PROFILE(Temp_Profile,t,i)
{
  real xx[ND_ND];
  real x;
  real y;
  face_t f;
  begin_f_loop(f,t)
  {
    F_CENTROID(xx,f,t);
    x=xx[0];
    y=xx[1];
    F_PROFILE(f,t,i)=273.2+324.564-5.97*y-401.153*x-
      56.529*x*y-1050.0*y*y-2213.0*x*x;
```

$n$	$y$	$x$	$T$ ( $C$ )
1	-0.1172	-0.183109	305.8
2	-0.08929	-0.183109	316.1
3	-0.056137	-0.183109	321.8
4	-0.025278	-0.183109	322.6
5	0.0	-0.183109	325.0
6	0.028889	-0.183109	325.1
7	0.063031	-0.183109	321.9
8	0.093562	-0.183109	317.4
9	0.119824	-0.183109	305.5
10	-0.093889	-0.129948	329.3
11	-0.059748	-0.126010	333.4
12	-0.030202	-0.122401	334.8
13	0.0	-0.119447	336.7
14	0.033157	-0.122401	335.1
15	0.066970	-0.127979	332.6
16	0.093233	-0.130933	328.4
17	-0.121138	-0.127651	322.9
18	-0.106365	-0.123713	334.1
19	-0.074521	-0.114525	338.7
20	-0.043990	-0.106978	342.1
21	0.0	-0.101399	334.7
22	0.045960	-0.107634	345.8
23	0.076491	-0.114197	343.7
24	0.104723	-0.123713	330.7
25	0.121794	-0.127323	321.4
26	0.0	-0.054473	347.2
27	0.0	-0.020345	327.7

Figure 2.19: List of measured points and their respective temperatures imported in Mathcad

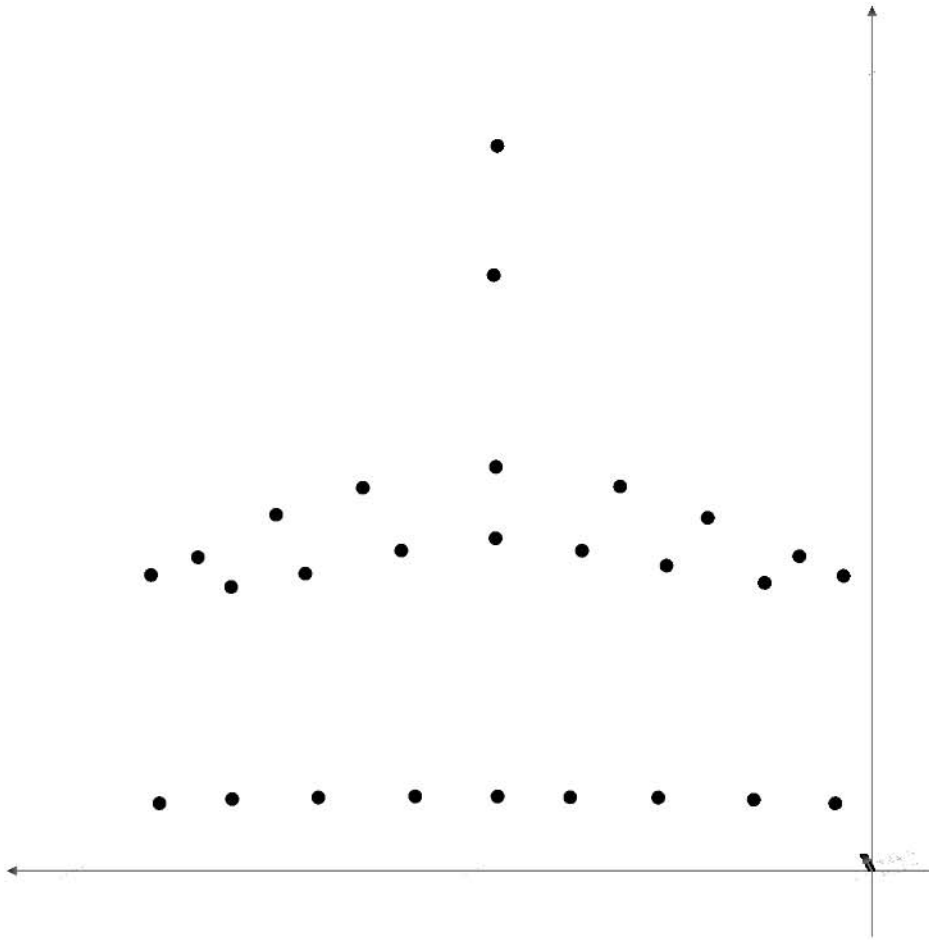


Figure 2.20: Position of the imported point into Mathcad as seen normal to the xy-plane



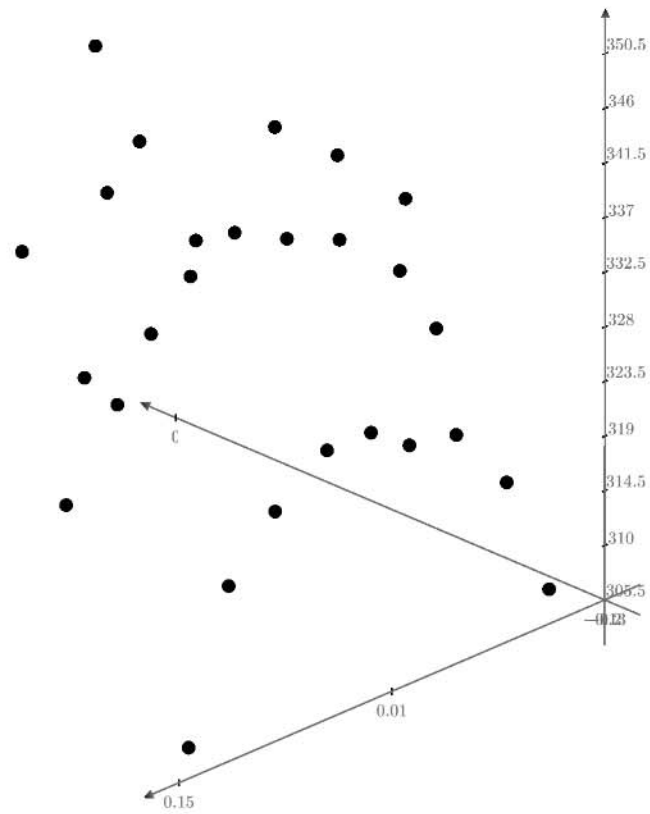
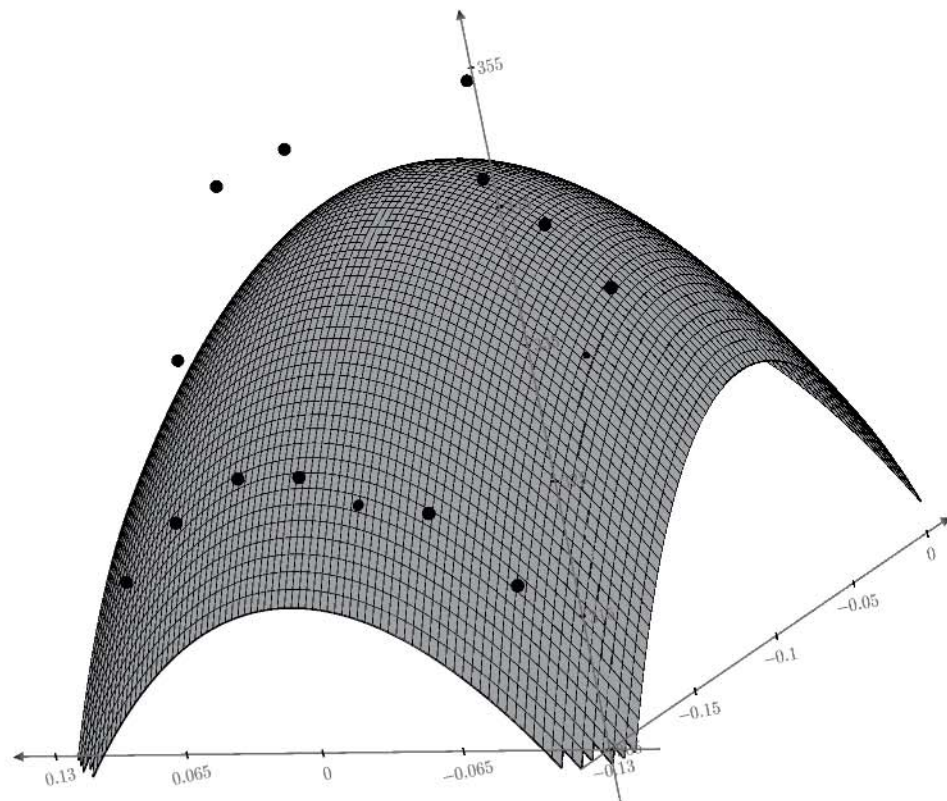


Figure 2.21: Position of the imported point into Mathcad as seen from a 45 degrees (the vertical axis is the temperature)

$$F^{(1)} = \begin{bmatrix} \text{"Coefficient"} \\ 324.564 \text{ C} \\ -401.153 \text{ C} \\ -5.97 \text{ C} \\ -56.529 \text{ C} \\ -2.213 \cdot 10^3 \text{ C} \\ -1.05 \cdot 10^3 \text{ C} \end{bmatrix}$$

$$\text{Temperature}(xx, yy) := F_{1,1} + F_{2,1} \cdot xx + F_{3,1} \cdot yy + F_{4,1} \cdot xx \cdot yy + F_{5,1} \cdot xx^2 + F_{6,1} \cdot yy^2$$



Temperature (C)

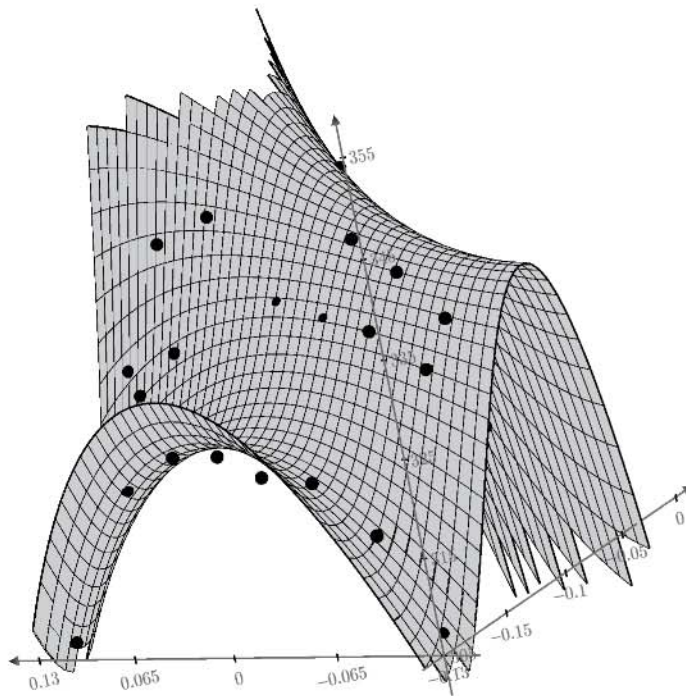
Figure 2.22: Second order regression of the temperature data in Mathcad

$$F^{(1)} = \begin{bmatrix} \text{"Coefficient"} \\ 302.241 \text{ C} \\ -1.549 \cdot 10^3 \text{ C} \\ 638.76 \text{ C} \\ (8.55 \cdot 10^3) \text{ C} \\ -1.567 \cdot 10^4 \text{ C} \\ (1.388 \cdot 10^3) \text{ C} \\ (2.837 \cdot 10^4) \text{ C} \\ (1.524 \cdot 10^4) \text{ C} \\ -4.33 \cdot 10^4 \text{ C} \\ -1.515 \cdot 10^3 \text{ C} \end{bmatrix}$$

$$T3rd(xx, yy) := F_{7,1} \cdot xx^2 \cdot yy + F_{8,1} \cdot xx \cdot yy^2 + F_{9,1} \cdot xx^3 + F_{10,1} \cdot yy^3$$

$$T2nd(xx, yy) := F_{1,1} + F_{2,1} \cdot xx + F_{3,1} \cdot yy + F_{4,1} \cdot xx \cdot yy + F_{5,1} \cdot xx^2 + F_{6,1} \cdot yy^2$$

$$\text{Temperature}(xx, yy) := T2nd(xx, yy) + T3rd(xx, yy)$$



Temperature (C)

Figure 2.23: Third order regression of the temperature data in Mathcad

$$F^{(1)} = \begin{bmatrix} \text{"Coefficient"} \\ 283.91 \text{ } C \\ -3.039 \cdot 10^3 \text{ } C \\ (4.571 \cdot 10^3) \text{ } C \\ (9.649 \cdot 10^4) \text{ } C \\ -4.975 \cdot 10^4 \text{ } C \\ (3.731 \cdot 10^4) \text{ } C \\ (6.705 \cdot 10^5) \text{ } C \\ (4.889 \cdot 10^5) \text{ } C \\ -3.195 \cdot 10^5 \text{ } C \\ -1.696 \cdot 10^3 \text{ } C \\ (1.543 \cdot 10^6) \text{ } C \\ (1.525 \cdot 10^6) \text{ } C \\ -3.165 \cdot 10^3 \text{ } C \\ -7.2 \cdot 10^5 \text{ } C \\ -6.087 \cdot 10^4 \text{ } C \end{bmatrix}$$

$$T3rd(xx, yy) := F_{7,1} \cdot xx^2 \cdot yy + F_{8,1} \cdot xx \cdot yy^2 + F_{9,1} \cdot xx^3 + F_{10,1} \cdot yy^3$$

$$T4th(xx, yy) := F_{11,1} \cdot xx^2 \cdot yy^2 + F_{12,1} \cdot xx^3 \cdot yy + F_{13,1} \cdot xx \cdot yy^3 + F_{14,1} \cdot xx^4 + F_{15,1} \cdot yy^4$$

$$T2nd(xx, yy) := F_{1,1} + F_{2,1} \cdot xx + F_{3,1} \cdot yy + F_{4,1} \cdot xx \cdot yy + F_{5,1} \cdot xx^2 + F_{6,1} \cdot yy^2$$

$$Temperature(xx, yy) := T2nd(xx, yy) + T3rd(xx, yy) + T4th(xx, yy)$$

Figure 2.24: Parameters for the fourth order regression of the temperature data in Mathcad

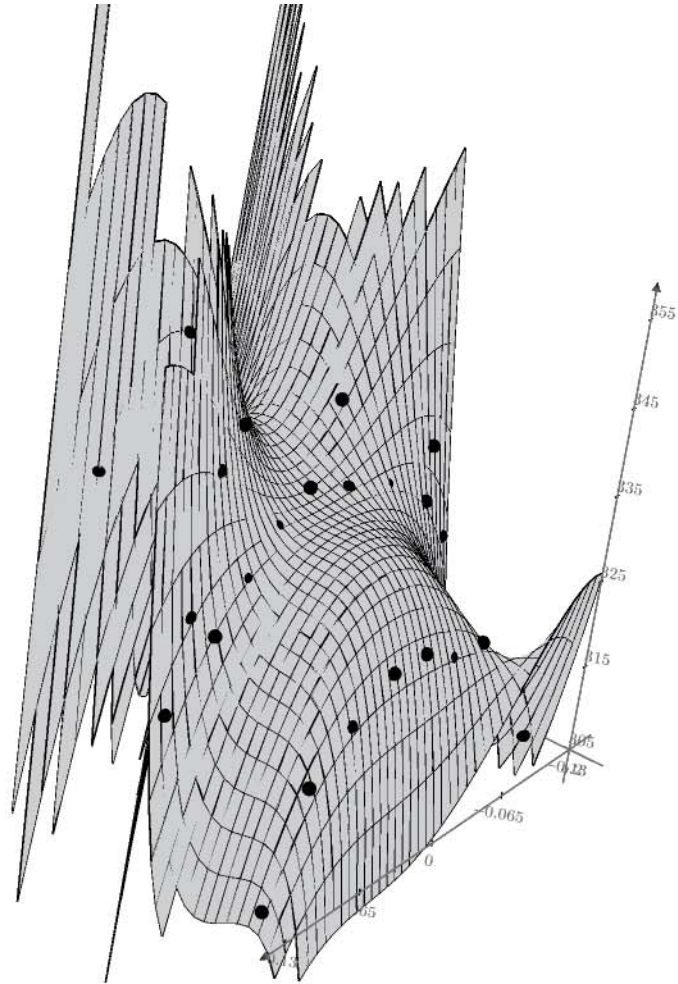


Figure 2.25: Surface plot of the fourth order regression of the temperature data in Mathcad

```

}
end_f_loop(f,t)
}

```

This UDF uses the position of each cell and assigns a temperature to that cell. When attached to a surface for boundary condition, the temperature of that surface will be calculated at the beginning of the simulation. Figure 2.26 shows the applied temperature as a boundary condition on the surface of the die. One important assumption made here is that the same temperature contour is applied to both the top and bottom of the die. This assumption may not be true if the bottom and top halves of the die have completely different temperature control system.

The phenomenon that was observed during the extrusion of LCPs from the coat-hanger die was the higher velocity of material coming out of the middle of the die. This phenomenon causes the material to have ripples in the middle and be stretched in the corners. Figure 2.27 shows how higher velocity of the material in the middle of the die affects the extruded film. To model this phenomenon and ideally change the design of the flow channel to have a uniform velocity at the lip of the die, the effect of the temperature on the viscosity of the LCP had to be considered. As can be seen in Figure 2.28, the velocity of the extrudate reduces about  $1\text{cm/s}$  from the middle of the die to the end of the die. This graph shows the velocity profile at the lip of the die from the middle to the end.

Figure 2.29 shows the velocity vectors for when the effect of temperature on the viscosity is considered. Higher temperature in the middle part of the die has reduced the viscosity from the design point. The coat hanger die is designed to operate in an isotropic condition and this simulation shows that a better temperature control system is needed to keep the temperature of the die constant. This can be done using additional heating cartridges close to the corners of the die lip. It should be noted that in this simulation only one half of the die is modeled. In the case of a big difference in the temperature distribution between the left and right half of the die, all the die should be modeled.

### 2.3 Rheological modeling on the structured and unstructured meshes for the directionality modeling

For this part of the simulation, ANSYS® DesignModeler is used for making the geometry of the flow domain and meshing is done using the ANSYS® Meshing software. Since the same mesh generated for solving the rheology is also used to model the directionality, the

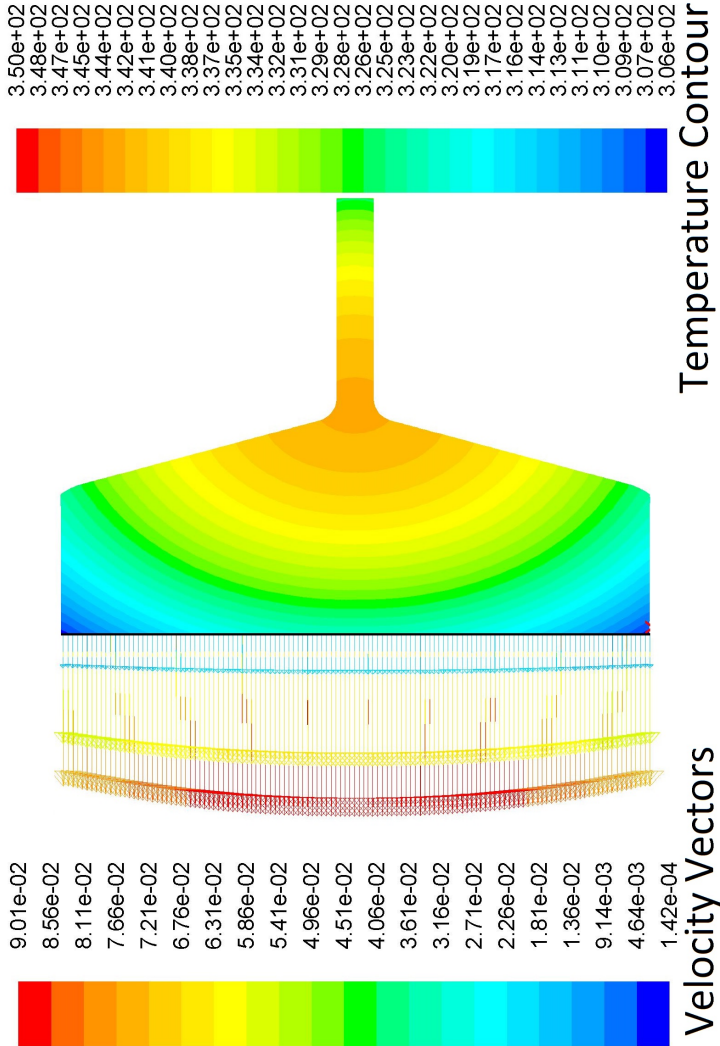


Figure 2.26: Temperature boundary condition from the measured data

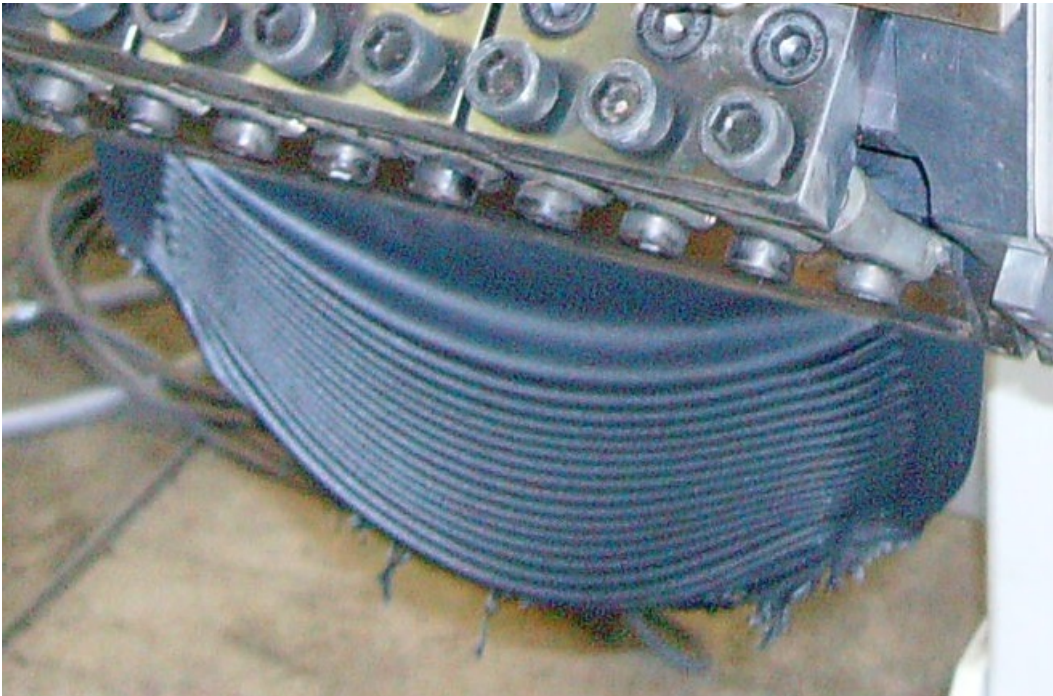


Figure 2.27: Film extrusion showing higher velocity at the middle of the die

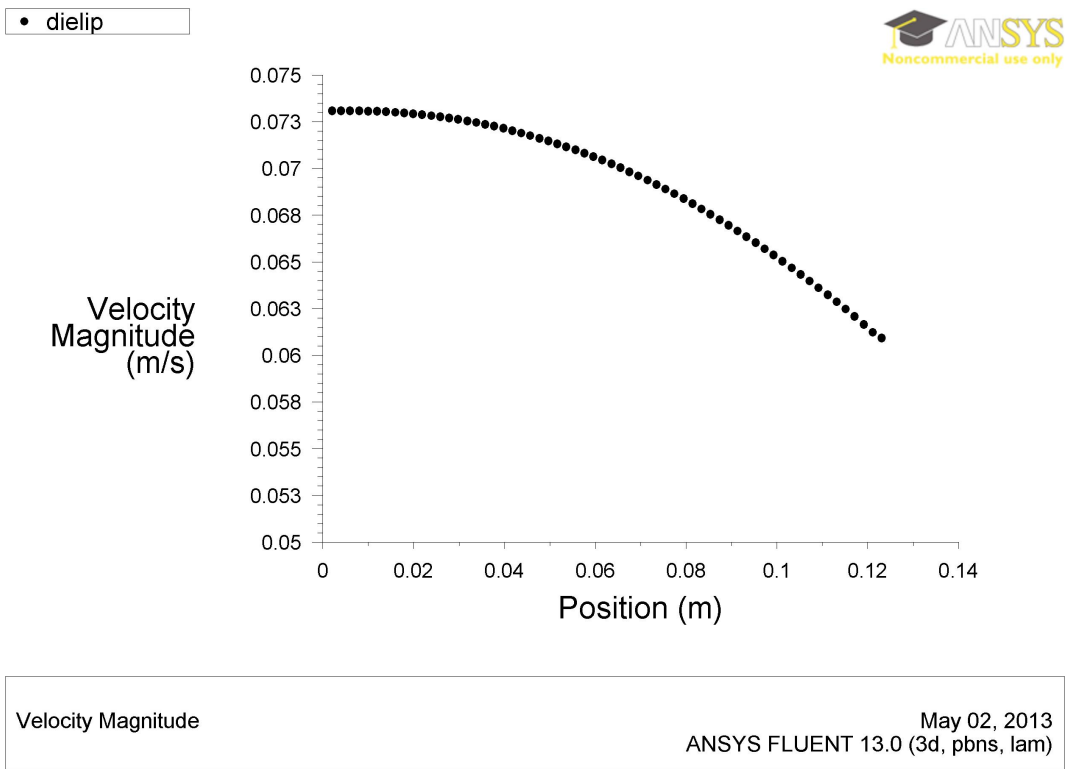


Figure 2.28: Velocity profile at the lip of the die (from the center to the edge)



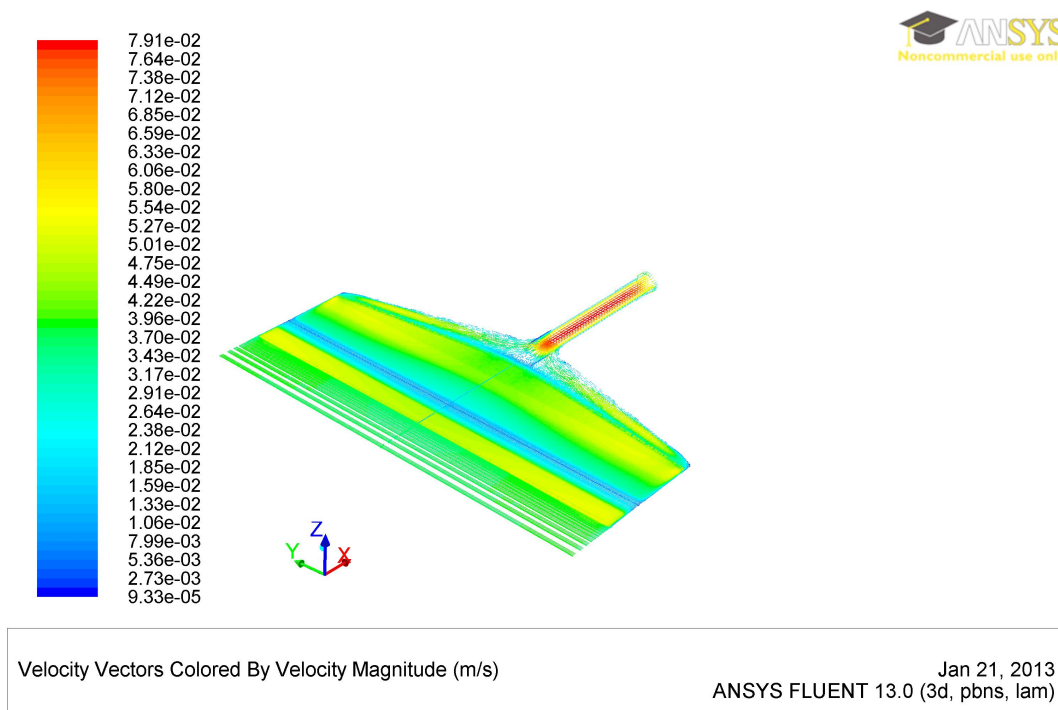


Figure 2.29: Velocity Vectors at the lip of the die

minimum size of the mesh should consider the minimum volume with aligned crystals. The cell volume in the work of Goldbeck-Wood et al. [22] is considered to be  $(100\text{nm})^3$ . At this scale it is possible to ignore the molecular entropic term and only consider the elastic torque applied to directors. Previous researchers' calculations with this method considered mostly a bulk of fluid with structured quadrilateral mesh elements (ex. Lavine and Windle [23], Goldbeck-Wood et al. [22]). Moreover, the geometry of the flow domain was defined to be a cube. As it is described later, the method developed here is able to account for complex geometries and is not restricted to one type of mesh. As a result, it is possible to use suitable shapes of elements with arbitrary orientation in an unstructured mesh for simulating the rheology. To demonstrate the ability of the code to handle different mesh structures, the geometry of the domain is meshed with different element types and the results are compared. Here the rheology of the polymer is simulated using *ANSYS® FLUENT® Release 13.0.0*. ANSYS® FLUENT® is a finite volume based flow solver that can simulate the flow on a wide variety of meshes. This ability makes it possible to simulate the flow in complex geometries and allows the use of different types of meshes in one simulation. To model the rheology of the polymer, different rheological models have been tried. Based on the die swell data and existing experimental measurements described in Ahmadzadegan et al. [24],

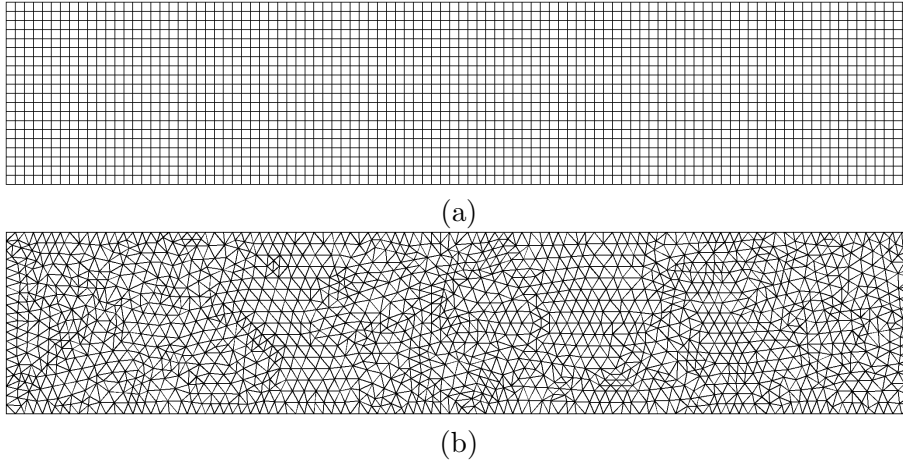


Figure 2.30: (a) Structured and (b) Unstructured meshes

the Power-law model described in FLUENT [21] is chosen here. Equation 2.4 describes the relationship between stress and strain in this model.

$$\tau = k\dot{\gamma}^n \quad (2.4)$$

In this equation,  $\tau$  is the stress tensor and  $\dot{\gamma}$  is the rate of deformation tensor.  $k$  and  $n$  are the flow consistency index and power-law index, respectively. The details of this choice are explained in Ahmadzadegan et al. [24]. In this study  $k = 1864 \frac{kg.s^{n-2}}{m}$  and  $n = 0.4651$ . After simulating the rheology, a user defined function (UDF) is coded to extract the rheological data from the solution and calculate its effect on the orientation of crystals.

As mentioned before, modeling rheology is done using *ANSYS® FLUENT® Release 13.0.0*. The geometry of the flow is modeled in a 2D channel and meshed with triangular and quadrilateral mesh for comparison. Since in this study modeling directionality is of prime interest and the geometry of the flow channel is very simple, the first order upwind scheme is used in the isothermal solver. The more accurate the rheology modeling, the more accurately one can predict the effect of rheology on the crystals. For modeling the fluid, capillary rheometer data using a LCR 6000 capillary melt rheometer is used for a LCP material in  $350^\circ C$  temperature. Capillary rheometry data shows a close match between the stress-strain curve and the power-law model and as a result, the power-law model with flow consistency index of  $k = 1864 Pa.s^n$  and a power-law index of  $n = 0.4651$  is considered.

Figure 2.30 shows the structured and unstructured mesh used for this study.

## 2.4 Die Swell Measurements

### 2.4.1 Numerical Simulation

ANSYS<sup>®</sup> POLYFLOW<sup>®</sup> can simulate the free surfaces of the extrudate after exiting the die which makes it ideal for simulating the die swell. There are two boundary conditions that needs to be satisfied for a successful simulation of the free surfaces in a steady state flow [25]. First, the normal velocity of the surface should be equals to zero.

$$\mathbf{v} \cdot \mathbf{n} = 0 \quad (2.5)$$

In this equation,  $\mathbf{v}$  is the velocity vector and  $\mathbf{n}$  is the vector normal to the surface. The condition represented by equation (2.5) is known as the kinematic condition. Second boundary condition on the surface of the extrudate is known as the dynamic condition which indicates there is no normal force on the surface.

$$\mathbf{f} = 0 \quad (2.6)$$

Outside the die the position of the surface of the extrudate is also unknown and needs to be calculated as an output of the simulation.

Since the numerical simulation here is going to be validated using the experimental results presented later in the design of the extrusion die section, the same geometry that is used in the experiment is used. The die is designed such that flow of the LCP is happening at high Weissenberg numbers ( $We$ ). This means that the elasticity of the material is important and should be taken into account. There are several viscoelastic models available in ANSYS<sup>®</sup>POLYFLOW<sup>®</sup>. A method of choosing between these viscoelastic models is to use the shear viscosity vs. shear rate curve. If at a typical shear rate of the flow through the die the viscosity is constant, then the Maxwell model or the Oldroyd-B is recommended. But, if the fluid shows shear thinning around this shear rate, then the Phan Thien-Tanner (PPT) or Giesekus model is recommended. The shear viscosity vs. shear rate curve for the LCP is shown in Figure 2.31. This data is from a test using *LCR6000* capillary melt rheometer. The shear rate at the wall,  $\dot{\gamma}_w$ , during extrusion changes from  $350 \text{ s}^{-1}$  to  $5200 \text{ s}^{-1}$ , which is in the shear thinning section of the curve in Figure 2.31. Based on this calculation, the PTT or Giesekus models are more appropriate to simulate the rheology inside the die. The single mode model is used here and the relaxation time is of the order of  $\frac{1}{\dot{\gamma}_w}$ .

Since LCPs are known to have lower viscosities compared to conventional polymers, it is rational to consider if the inertia of the fluid is important during the flow. Inertia effects

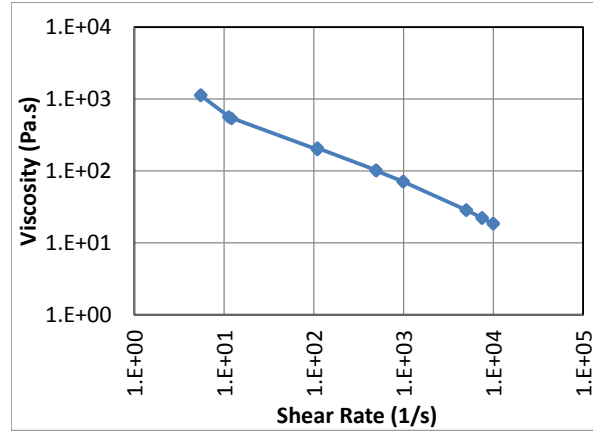


Figure 2.31: Shear viscosity vs. Shear rate for LCP at  $T = 350^{\circ}C$

are characterized by the *Reynolds* number.

$$Re = \frac{\rho V D}{\eta} \quad (2.7)$$

The  $Re$  of the flow in the die changes between  $Re \sim 0.02$  to  $Re \sim 4$ .  $Re$  is even lower for PP due to its higher viscosity. This means that the viscous effects are dominant compared to the inertia effects. To check the importance of inertia effect as compared to the elastic effect, a *Mach* number is used as follows;

$$M = \sqrt{We.Re} \quad (2.8)$$

When the effect of inertia is important, the flow field is both affected by inertia and viscoelastic forces. This adds more nonlinearity to the simulation and causes convergence difficulties. In this study, since the *Mach* number is small and none of the inertia effects has been observed in the experimental measurement (i.e. delayed die swell), the effect of inertia is neglected. This can be done by choosing the value of zero for the density of the polymer and not considering the inertia term. Ignoring inertia terms remove some of the nonlinearity and also makes the calculation cheaper computationally.

Figure 2.32 shows the mesh used to simulate the flow inside and outside the capillary die. Inside the flow channel the solver needs to solve the Navier-Stokes equations in addition to the energy equation. On the other hand, in the body of the die, only the conduction heat transfer exists and the quality of the mesh is less important. Based on this consideration, inside the flow channel, mostly structured mesh was generated with the consideration of

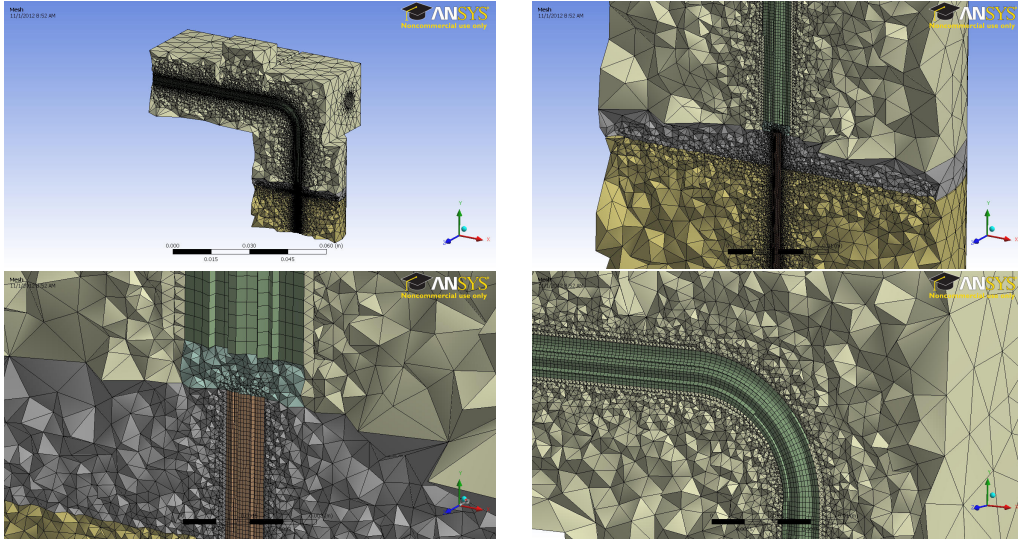


Figure 2.32: Generated mesh on the capillary die

the streamlines and outside the flow channel unstructured mesh was generated to reduce the number of mesh and computational cost of the simulation.

#### 2.4.1.1 Phan Thien-Tanner Viscoelastic Model

Since in this case, elasticity of the polymer is important in the flow through the capillary, a viscoelastic model needs to be used to model the rheology. For incompressible and isothermal flows the momentum and continuity equations are as follows:

$$\nabla \cdot \mathbf{v} = 0 \quad (2.9)$$

$$-\nabla p + \nabla \cdot \mathbf{T} + \mathbf{f} = \rho \mathbf{a} \quad (2.10)$$

In these equations  $\mathbf{v}$  is velocity,  $\mathbf{a}$  is acceleration,  $\mathbf{f}$  is volume force and  $\mathbf{T}$  is the extra-stress tensor. For viscoelastic fluids, the extra-stress tensor is composed of a viscoelastic  $\mathbf{T}_1$  and a purely viscous  $\mathbf{T}_2$  components.

$$\mathbf{T} = \mathbf{T}_1 + \mathbf{T}_2 \quad (2.11)$$

The stress tensor is coupled to the momentum and continuity. The purely viscous part of the extra-stress tensor,  $\mathbf{T}_2$ , is optional but recommended for viscoelastic fluid. It helps with stability of the solution and is defined as:

$$\mathbf{T}_2 = 2\eta_2\mathbf{D} \quad (2.12)$$

where  $\mathbf{D}$  is the rate of deformation tensor and  $\eta_2$  is the viscosity for the Newtonian component. In ANSYS<sup>®</sup> POLYFLOW<sup>®</sup>, a viscosity ratio,  $\eta_r$ , defined as  $\eta_2/\eta$  relates the viscosities as

$$\eta_1 = (1 - \eta_r)\eta \quad (2.13)$$

and

$$\eta_2 = \eta_r\eta \quad (2.14)$$

where,  $\eta_1$  is the viscoelastic coefficient of the total viscosity  $\eta$ . In ANSYS<sup>®</sup> POLYFLOW<sup>®</sup>, the equations relating the stress, velocity, pressure and moving boundaries are coupled and the system of equations are solved using a full Newton-Raphson scheme. The constitutive equation for the stress  $\mathbf{T}_1$  is calculated from the following differential equation:

$$g(\mathbf{T}_1) \cdot \mathbf{T}_1 + \lambda \frac{\delta \mathbf{T}_1}{\delta t} = 2\eta_1\mathbf{D} \quad (2.15)$$

The relaxation time  $\lambda$  is defined as the required time for the shear stress to reduce to about 1/3 of its equilibrium value when the strain rate vanishes. For Newtonian flows the relaxation time is zero and high values for the relaxation time increase the memory retention.

The term  $\delta \mathbf{T}/\delta t$  is an objective derivative defined as a linear combination of lower and upper-convected derivatives

$$\frac{\delta \mathbf{T}}{\delta t} = \frac{\xi}{2} \overset{\Delta}{\mathbf{T}}_1 + \left(1 - \frac{\xi}{2}\right) \overset{\nabla}{\mathbf{T}}_1 \quad (2.16)$$

for  $0 \leq \xi \leq 2$ .  $\overset{\Delta}{\mathbf{T}}_1$  is the lower-convected time derivative of  $\mathbf{T}_1$  and  $\overset{\nabla}{\mathbf{T}}_1$  is the upper-convected time derivative of  $\mathbf{T}_1$ .

$$\overset{\Delta}{\mathbf{T}}_1 = \frac{D\mathbf{T}_1}{Dt} + \mathbf{T}_1 \cdot \nabla \mathbf{v}^T + \nabla \mathbf{v} \cdot \mathbf{T}_1 \quad (2.17)$$

$$\overset{\nabla}{\mathbf{T}}_1 = \frac{D\mathbf{T}_1}{Dt} - \mathbf{T}_1 \cdot \nabla \mathbf{v} + \nabla \mathbf{v}^T \cdot \mathbf{T}_1 \quad (2.18)$$

In case of the Phan-Thien-Tanner model [26],  $\mathbf{T}_1$  is calculated from

$$\exp\left[\frac{\epsilon\lambda}{\eta_1}\mathbf{T}_1\right]\mathbf{T}_1 + \lambda\left[\left(1 - \frac{\xi}{2}\right)\overset{\nabla}{\mathbf{T}}_1 + \frac{\xi}{2}\overset{\Delta}{\mathbf{T}}_1\right] = 2\eta_1\mathbf{D} \quad (2.19)$$

and  $\mathbf{T}_2$  is calculated from equation 2.12. In this model,  $\xi$  and  $\epsilon$  are material properties of the polymer that control the shear viscosity and elongational viscosity, respectively. Defining a non-zero  $\epsilon$  causes a bounded steady extensional viscosity. An increasing  $\epsilon$  reduces or even cancels the strain hardening while  $\xi$  affects shear thinning properties as well as the amount of second normal stress differences. Strain hardening occurs when  $\epsilon$  is approximately between  $10^{-3}$  and  $10^{-2}$  and vanishes at  $\epsilon \approx 10^{-1}$ .

As can be seen there are several material properties to be defined for the PTT model. These properties are  $\xi$ ,  $\epsilon$ ,  $\eta_r$ ,  $\eta$  and  $\lambda$ . Since in this study the effects of the inertia is ignored, the density of the polymer is set to zero ( $\rho = 0$ ). As mentioned earlier, it is important to add a purely viscous term to the extra stress tensor for stability reasons. This is especially important for the PTT method and the ratio of the Newtonian viscosity,  $\eta_2$ , to the total viscosity,  $\eta$ , must be greater than or equal 1/9 ( $\eta_r \geq 1/9$ ).

In viscoelastic flows with high  $We$  numbers, it is crucial to use an evolution scheme on the volume flow rate or relaxation time. The start of the evolution should be with a  $We \leq 0.3$ . In the case of extrusion with free surface boundary conditions, it is also recommended to apply evolution to the moving boundaries. This technique progressively adds the effect of the kinematic condition into the system. For 2D viscoelastic flows with a single mode, it is possible to use the so-called 4x4-SU interpolation, which combines a high discretization level for the extra-stress tensor and the streamline upwinding method. This combination is robust for solving problems where elasticity plays a significant role, especially in the presence of flow singularities, such as die lips [25].

#### 2.4.1.2 Rheological Parameters

For finding the rheological parameters, the information presented in Figures 2.31 and 2.33 were used in POLYMAT. In POLYMAT it is possible to iterate and find the best rheological properties for a specific problem. There are many consideration to be taken into account for finding the right rheological properties. First of all, the experimental measurements of the shear viscosity (Fig. 2.31) and the storage and loss modulus (Fig. 2.33) needs to be imported to POLYMAT. Based on how important various rheological parameters are, it is possible to change the way the curve fitting will take place. For example in 2D extrusion, velocity rearrangement and normal stress differences affect the die swell. The velocity profile is basically the result of viscous forces and the normal stresses are the result

of viscoelasticity. The effect of strain thinning and strain hardening on the other hand is negligible in extrusion. For the best curve fitting inside POLYMAT, there are some changes done in the fitting parameters for the LCP as follows:

- Range of relaxation times:  $0.001 < \lambda < 0.1$
- Window of shear rates:  $0.1 < \dot{\gamma} < 11000$
- Window of frequencies:  $0.1 < f < 200$
- Weight of shear viscosity curves: 1.0
- Weight of  $G'$  and  $G''$  curves: 10

Based on these inputs, the fitted parameters for the LCP are

- $\eta = 4085 \text{ Pa}\cdot\text{s}$
- $\lambda = 0.316E - 02 \text{ s}$
- $\epsilon = 0.8603E - 02$
- $\xi = 0.8535$
- $\eta_r = 0.1712E - 02$

## 2.4.2 Experimental Measurements

To measure the die swell of polymers experimentally, two different extrusion dies with  $5\text{mm}$  and  $20\text{mm}$  land lengths were designed and built. Figures 2.34a and 2.34b show the two solid parts of the die. The extruder used in this experiment was a *DSM Xplore Micro 15cc twin screw compounder* (Figure 2.36). Using the upper part alone will give a die with land length of  $5\text{mm}$  and using the upper and lower parts together, will give a circular die with a land length of  $20\text{mm}$ . The two part attach together using four bolts from the bottom of the lower part. The flow channel in the case of  $20\text{mm}$  die will align perfectly to ensure a smooth channel without steps. Figure 2.35 shows a cross section of the melt pipe that guides the flow of polymer from the extruders to the die.



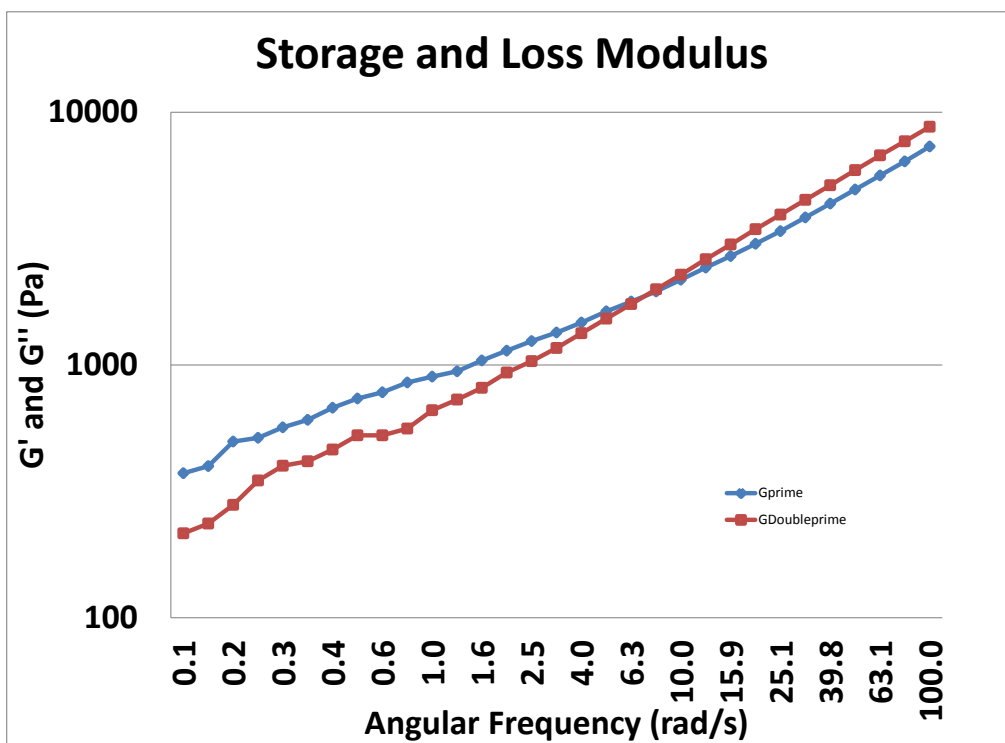


Figure 2.33: Change of elastic  $G'$  and Loss  $G''$  modulus with shear rate

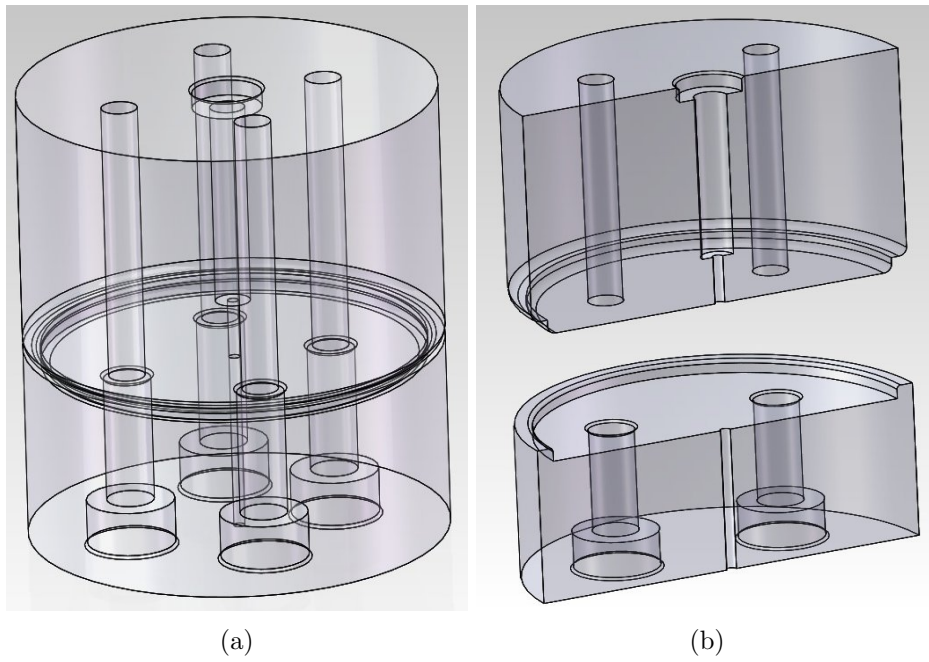


Figure 2.34: Die geometry (a) Wireframe (b) cross section

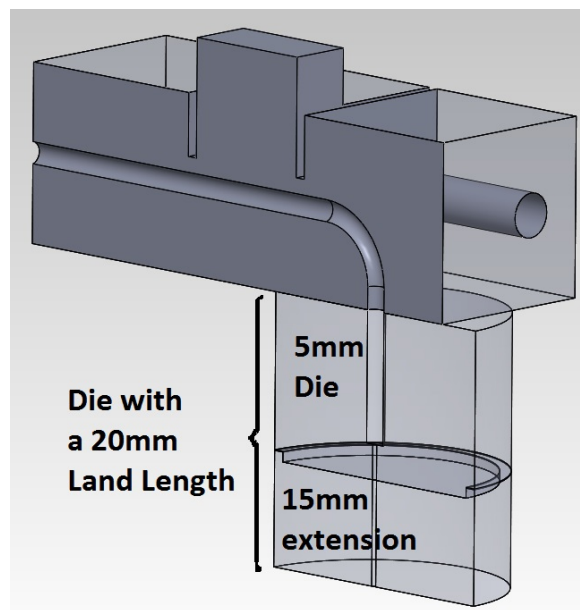


Figure 2.35: Melt pipe connecting the extruder to the die.



Figure 2.36: DSM Xplore Micro 15cc twin screw compounder

#### 2.4.2.1 ISO Standard for Extrudate Swelling

Based on Section 7.9 of the International Standard ISO-11443 [27], *Measurements of extrudate swelling*, die swell depends on many factors including:

- Test temperature
- Time since extrusion
- Manner of cooling
- Length of the extrudate
- Capillary die length
- Diameter and entry geometry
- Barrel diameter

and the results obtained can be very sensitive to the method and details of the experiment. As a result, the data on the extrudate swell is only comparable if all the testing conditions are identical.

There are two methods of measuring the die swell.

1. Measuring the diameter of the extrudate after it solidifies using micrometers

2. Using a photographic or optical method that does not involve any mechanical contact with the extrudate

In this study the second method is used. There are some important steps to be taken to eliminate the effect of the gravity on the die swell as follows:

- Any extrudate attached to the die should be removed as close as possible to the die
- Extrudate length at the time of the measurement should not be longer than 5cm
- Each measurement should be done at the same position and distance from the die lip
- To minimize the effect of the temperature drop on the extrudate, it is possible to use a confined temperature controlled chamber to extrude in

#### 2.4.2.2 Design of the Extrusion Die

A characteristic shear rate for axisymmetric flows can be considered as the shear rate at the wall of the die.

$$\dot{\gamma}_w = \frac{4Q}{\pi r^3} \quad (2.20)$$

The design of the extrusion die is done such that the extrusion happens when viscoelasticity is important. The importance of the viscoelasticity of a flow is measured using Weissenburg number ( $We$ ).  $We$  is defined as the product of relaxation time of the fluid and shear rate of the flow. It compares the elastic forces to the viscous effects.

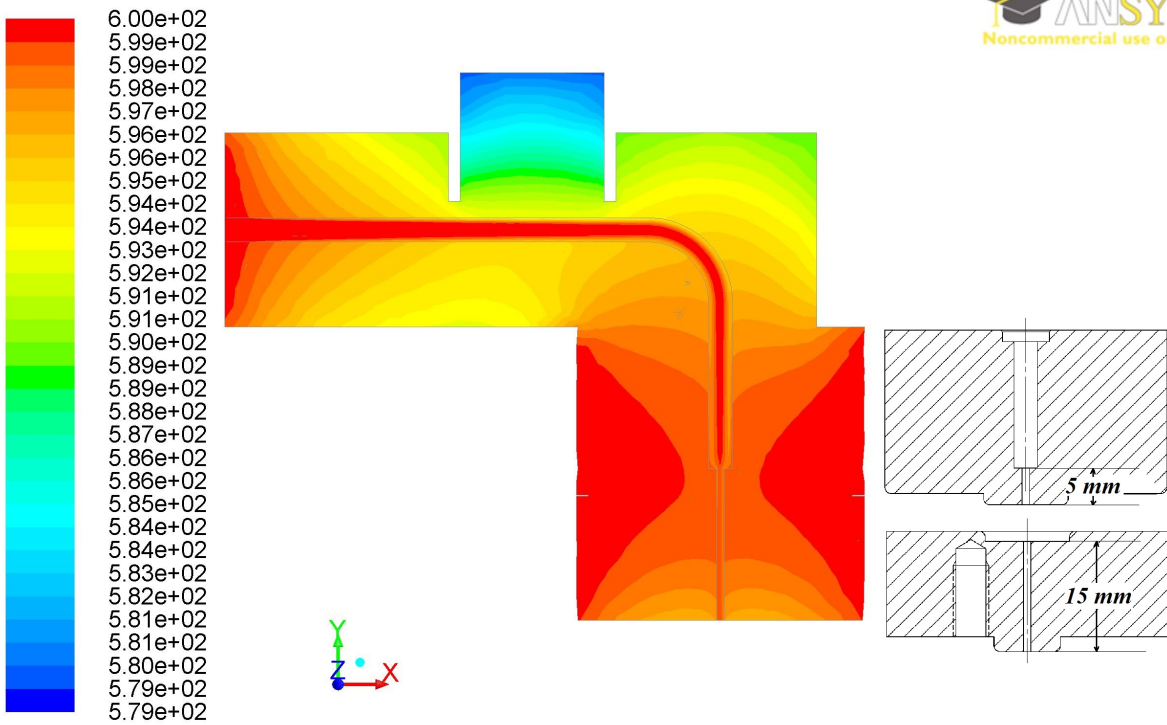
$$We = \lambda \times \dot{\gamma} \quad (2.21)$$

For a flow with  $We < 1$  the effect of viscosity is dominant and for  $We > 1$  the effect of the elastic forces are more important. An appropriate relaxation time in which the polymer changes from viscous to elastic can be found from the inverse of the frequency at which the elastic modulus,  $G'$ , and the loss modulus,  $G''$ , intersect. To obtain the  $G'$  and  $G''$  for LCP material, an oscillatory rheometer, AR2000, was used. The Figure 2.33 shows how  $G'$  and  $G''$  changes with the shear rate.

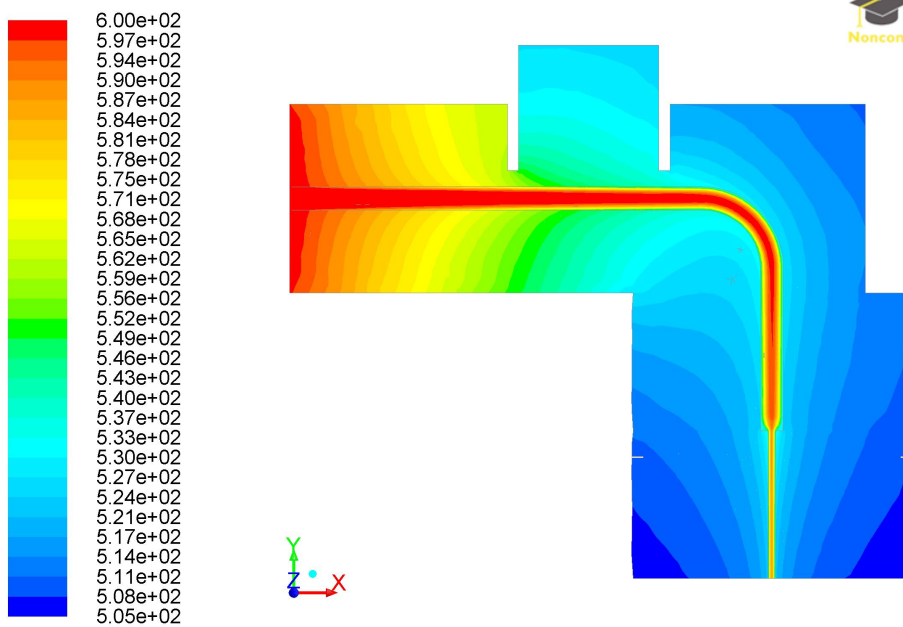
The mass flow rate of the polymer is chosen so that the  $We$  is always more than 1.  $We$  for the LCP changes from 60 to 550. To investigate the effect of the relaxation time of the polymer, two different land lengths were chosen for the die. The two land lengths were chosen so that the time spent inside the land length in some cases is less than the relaxation time and in other cases more than the relaxation time. Based on this rational,  $\Delta x = 5mm$  and  $\Delta x = 20mm$  were chosen.

The die was also thermally modeled in FLUENT® to determine the type and power of the heating elements needed to keep the flow isothermal. Figure 2.37 shows the contours of temperature in the middle surface of the die with and without the heating elements.

Figures 2.38 and 2.39 are the drawings of the two parts of the die. The part in Figure 2.38 has an abrupt contraction with the land length of 5mm. The part in Figure 2.39 attaches to the bottom of the the first part to form a die with a 20mm land-length. Two different heater bands are used to keep the temperature constant during the extrusion process.



(a)



Contours of Static Temperature (k)

Nov 01, 2012  
ANSYS FLUENT 13.0 (3d, pbns, lam)

(b)

Figure 2.37: Temperature profile (a)with and (b) without heating elements

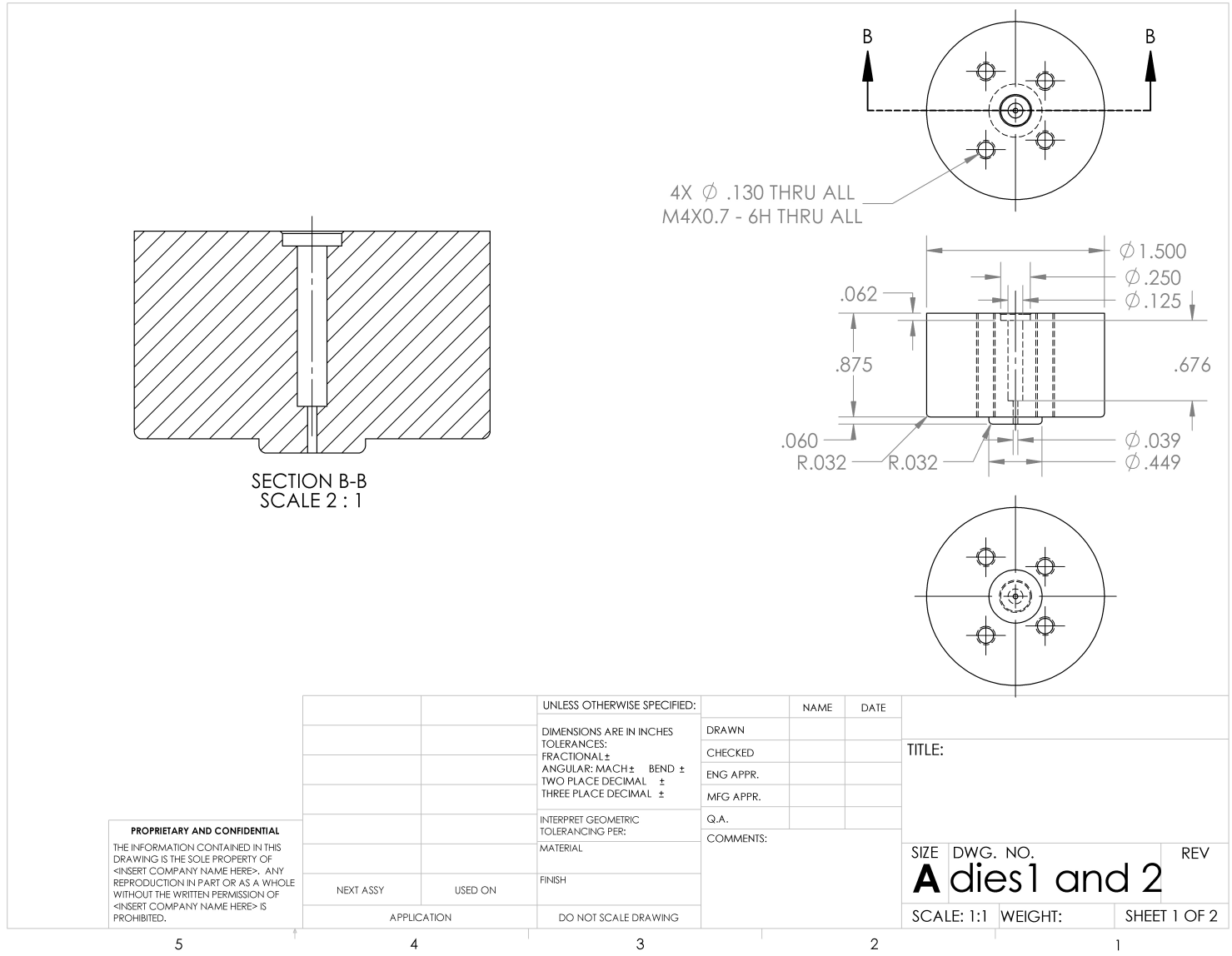


Figure 2.38: 5mm Capillary Die

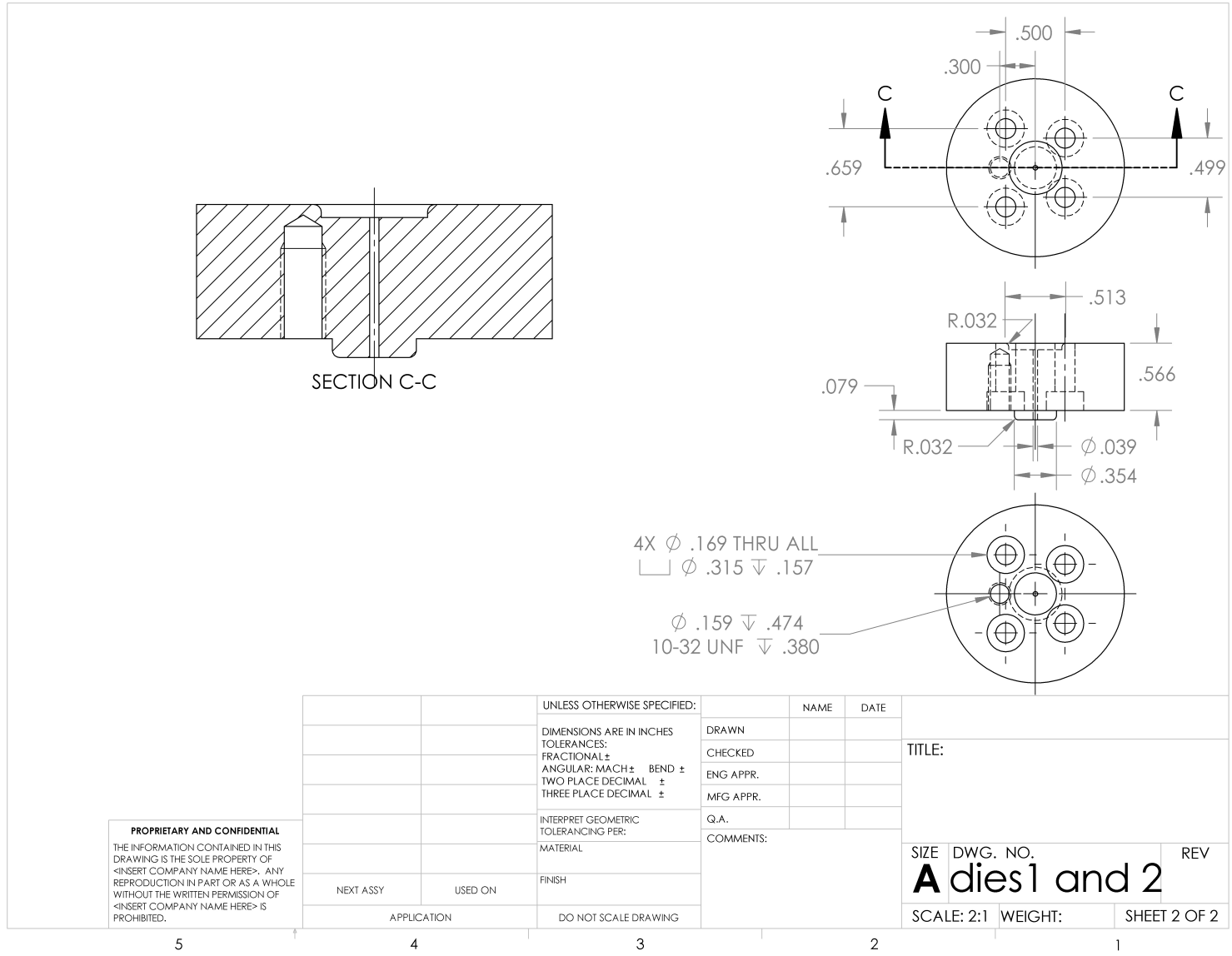


Figure 2.39: 15mm Capillary die attachment



Table 2.5: Temperature from the hopper to the lip of the die

	Zone 1 ( $^{\circ}C$ )	Zone 2 ( $^{\circ}C$ )	Zone 3 ( $^{\circ}C$ )	Lip of the Die ( $^{\circ}C$ )
LCP	330	340	350	350
PP	198	215	232	232

### 2.4.2.3 Experimental Procedure

To compare the die swell of the LCP with conventional polymers, the experiment was run using LCP and Polypropylene *Exxon PP1042*. PP was chosen because of the fact that both rheological and die swell experiments are available in the literature on this polymer, Huang and Tao [28].

A nine megapixel camera in macro mode was used to take the pictures and two telescopic lights were shined from two sides to the extrudate to be able to capture sharp pictures with maximum shutter speed. A metal measuring stick was placed as close as possible to the lip of the die for reference and image post-processing. In this experiment the extrudate is leaving the die at room temperature and solidifies. Figure 2.40 shows the setup of the experiment.

As can be seen in Figure 2.40, there is a heating element around the die and a thermostat close to the lip of the die to control the temperature. For PP the temperature at the lip is set to  $232^{\circ}C$  and for LCP to  $350^{\circ}C$ . It is very important to keep the temperature of the entrance of the screws well below the melting temperature of the polymer. This keeps the pellets solid and makes it possible for the polymer to easily enter the gaps between screws. Table 2.5 shows the gradient of the temperature from the hopper to the lip of the die for LCP and PP. These temperatures are shown for the zones in Figure 2.36 as well as the lip of the die.

The procedure for the experiment started with running a purging compound (a variation of HDPE) through the machine to clean the screws and the die. This step was done to clean the die from the left over polymer residue. After the die temperature is stabilized based on the temperatures in Table 2.5, LCP pellets was added to the hopper and run through the die with  $5mm$  land length. The rotational speed of the screws was changed for each land length and material. To find the mass flow rate of the flow for each rotational speed, the weight of the extrudate was measured during one minute. This measurement was done twice to reduce the error associated with the measurements. The four experimental graphs in Figure 2.43 shows how die swell is changing as a result of increasing the shear rate for PP and LCP. The die swell results are extracted from several high resolution pictures taken

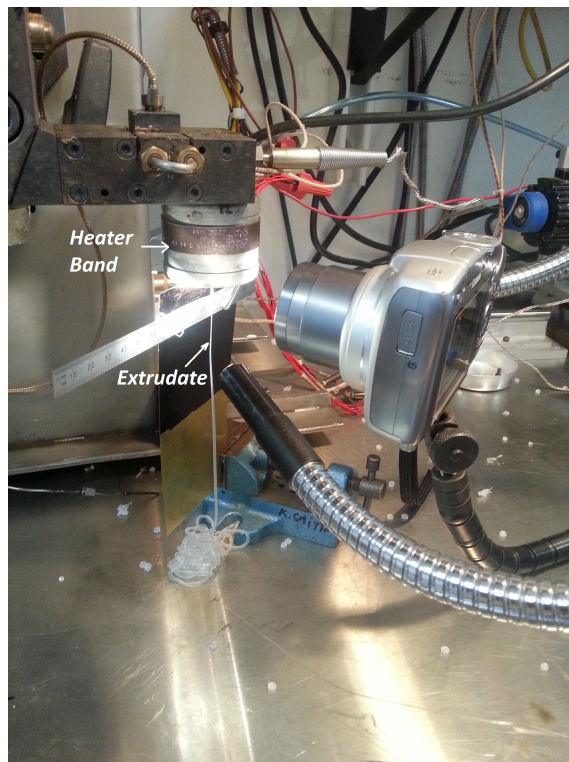


Figure 2.40: Setup of the experiment

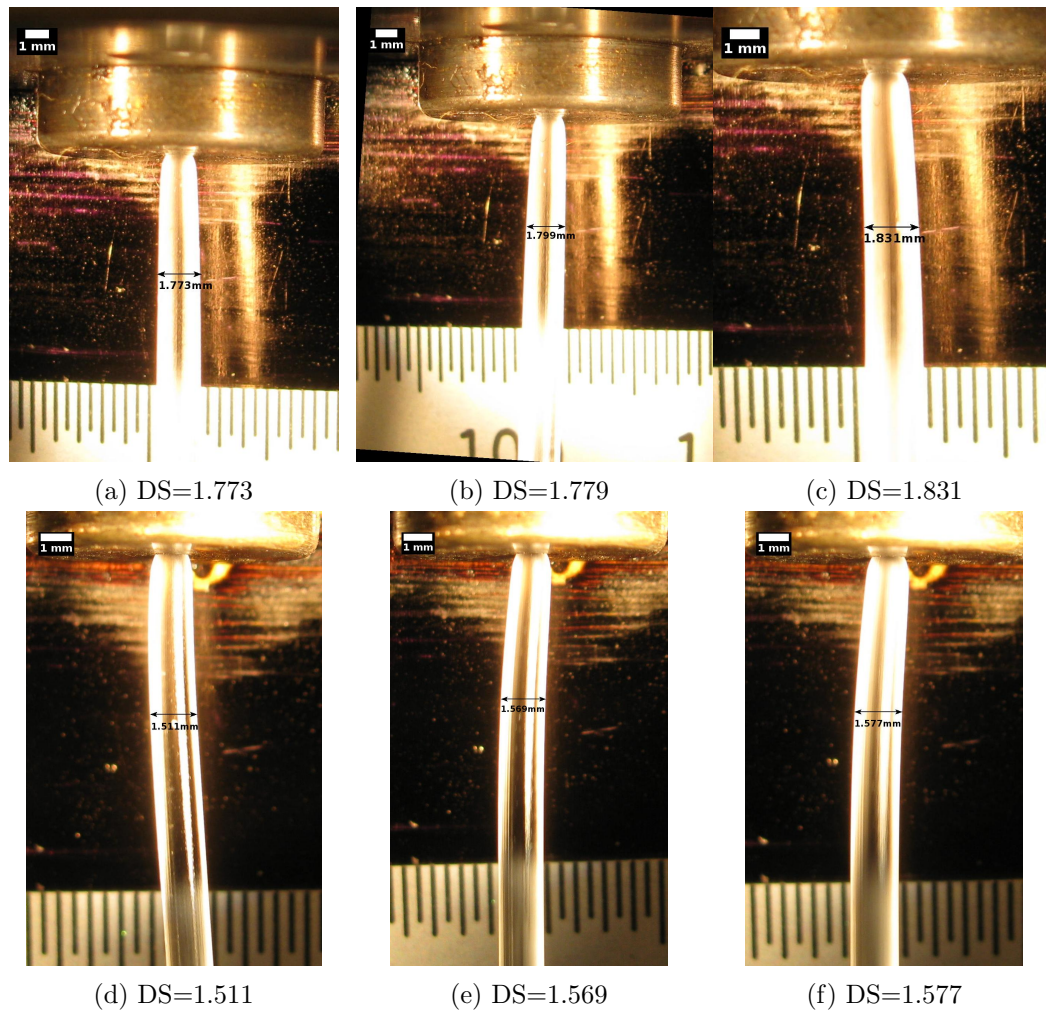


Figure 2.41: Die swell for PP at different shear rates.

during the extrusion. ImageJ image processing software was used later to calibrate, scale and measure the diameter of the extrudate as shown in Figures 2.41 and 2.42.

Table 2.6 lists the die swell measured with ImageJ for different average velocities. Average velocities are calculated based on the mass flow rate of the polymer. The density of the LCP is  $\rho = 1.6 \text{ g/cm}^3$  and the density of the PP is  $\rho = 0.9 \text{ g/cm}^3$ . The four graphs in Figure 2.43 shows how die swell is changing by increasing the shear rate for PP and LCP. As can be seen, the die swell for the LCP is very small and in some cases smaller than one (negative die swell).

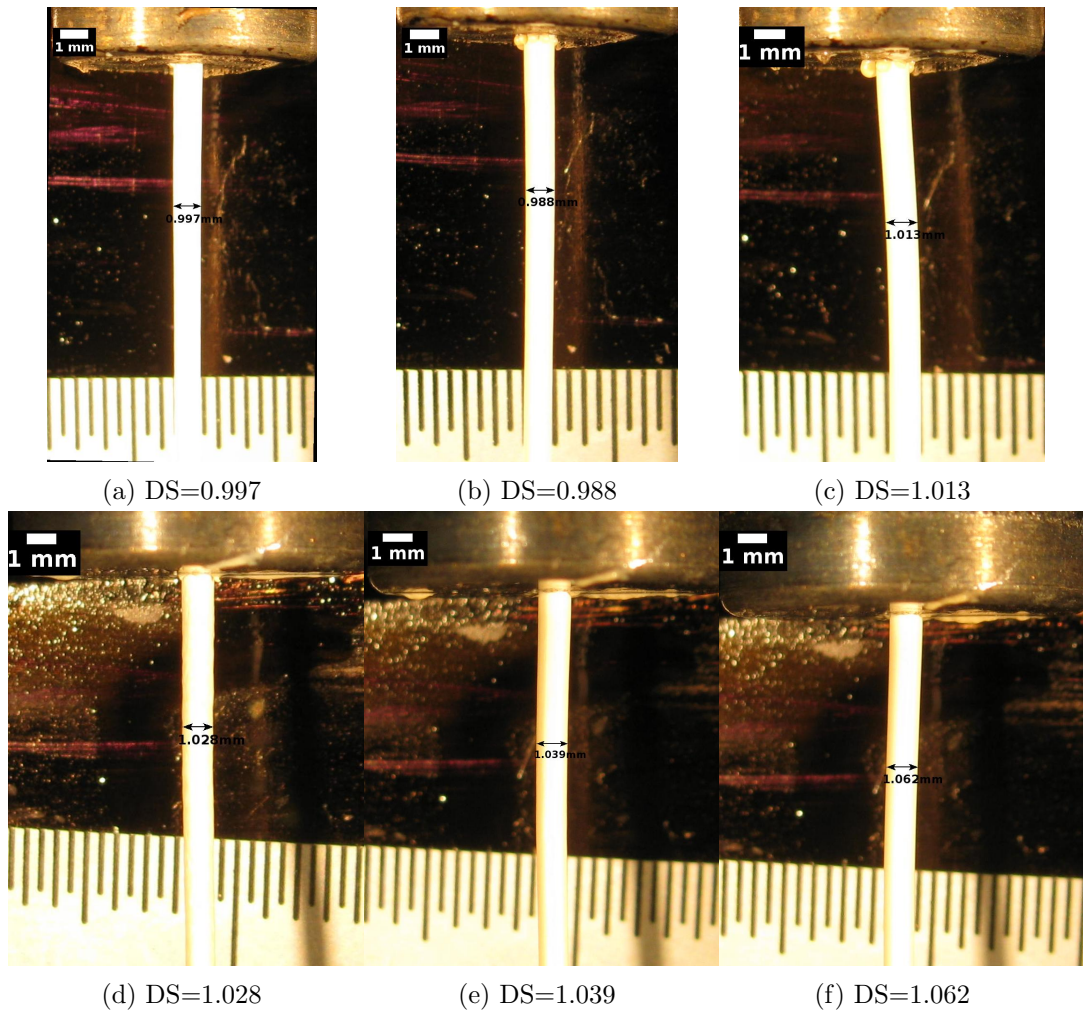


Figure 2.42: Die swell for LCP at different shear rates.

Table 2.6: Die swell ratio for PP and LCP for different average velocities and land lengths

	Land length (mm)	Average velocity (cm/s)	Experimental Die swell ratio
PP	5	3.174	1.773
		4.262	1.825
		5.698	1.799
		9.792	1.856
		12.147	1.831
	20	1.727	1.511
		2.916	1.53
		4.857	1.569
		6.102	1.579
		8.367	1.577
LCP	5	7.209	0.997
		13.403	0.994
		16.606	0.988
		22.747	0.981
		41.647	1.013
	20	5.816	1.028
		11.128	1.016
		16.599	1.039
		26.029	1.047
		50.467	1.062

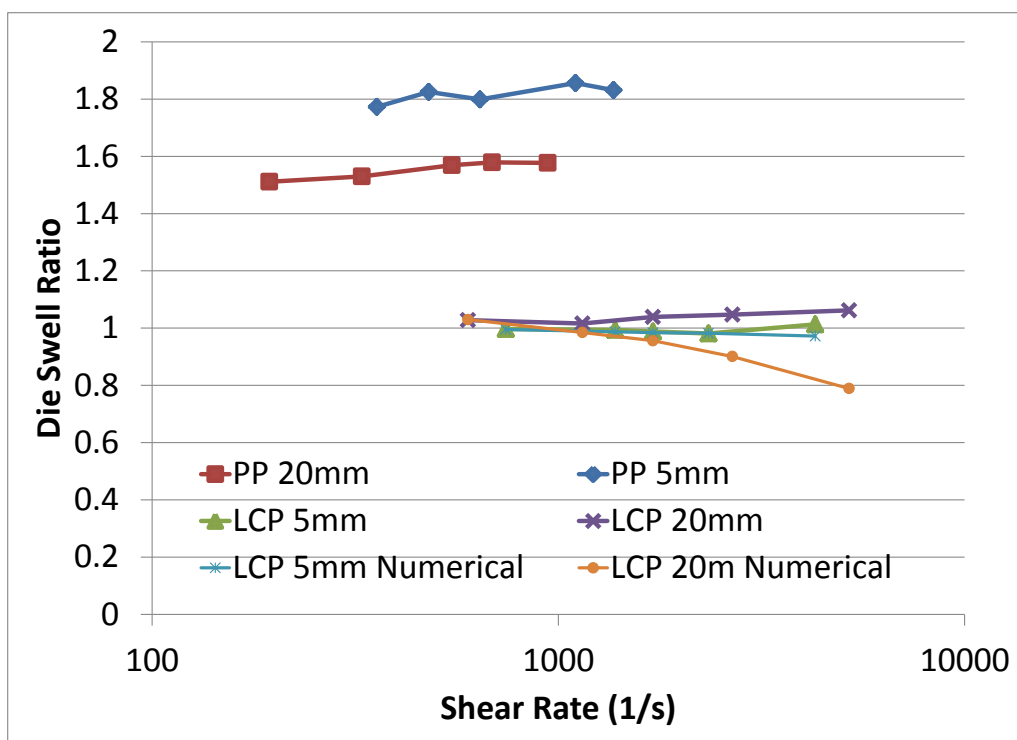


Figure 2.43: Comparison between die swell of LCP and PP for different land length and shear rate

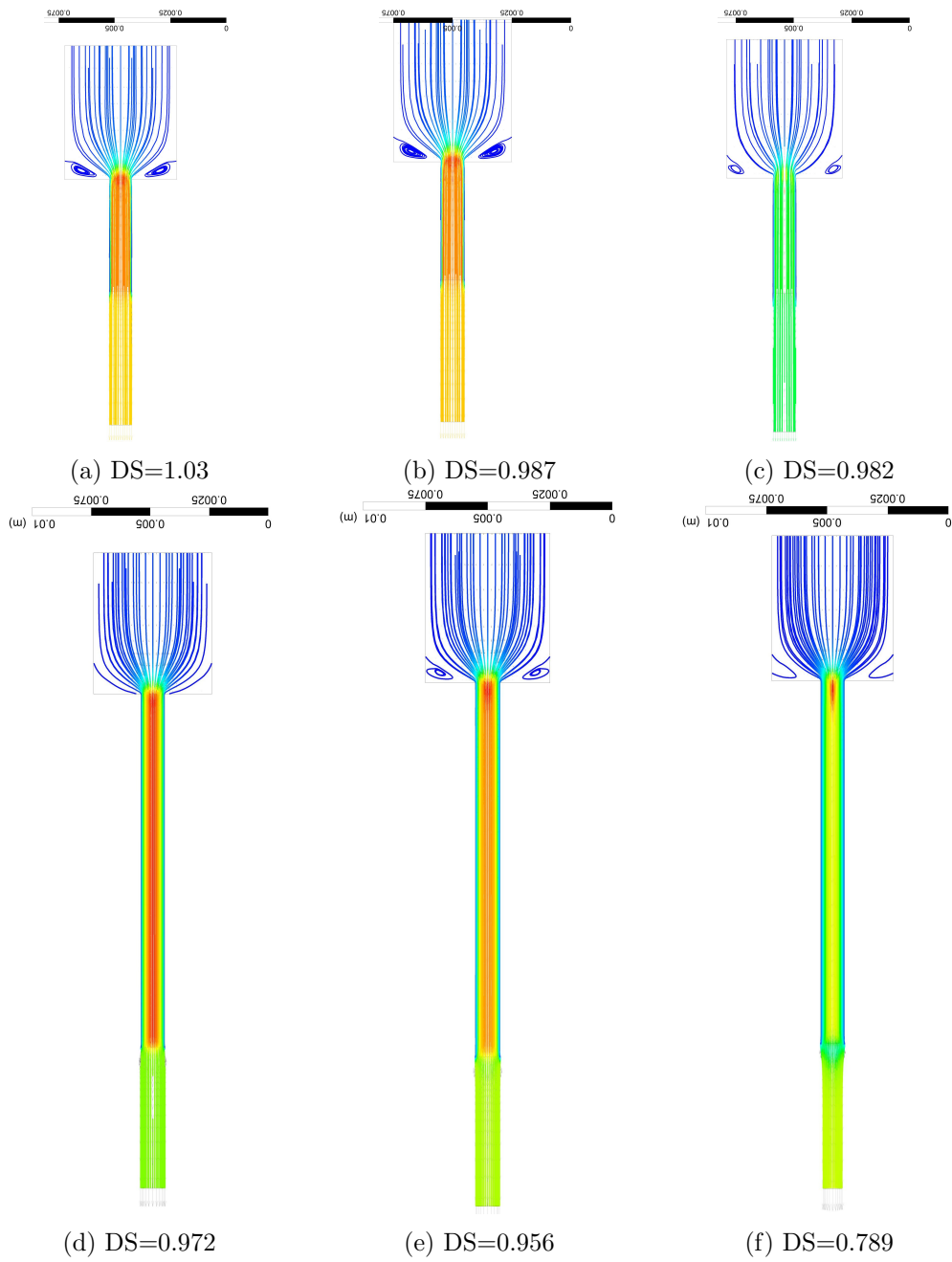


Figure 2.44: Velocity profile at the Lip of the Die

### 2.4.3 Results

Table 2.6 lists the die swell measured with ImageJ for different average velocities. Average velocities are calculated based on the mass flow rate of the polymer. The density of the LCP and PP are  $\rho_{LCP} = 1.6 \text{ g/cm}^3$  and  $\rho_{PP} = 0.9 \text{ g/cm}^3$ , respectively. As can be seen, the die swell for the LCP is very small and in some cases smaller than one (negative die swell). Figure 2.44 shows the streamlines inside the die for LCP material in both cases of 5mm and 20mm land lengths with the respective amount of die swell for each case. Comparing the numerical simulation of the die swell for the LCP to the experimental results show a good match at the shear rate that the material properties were obtained. As the shear rate increases, the simulation predicts increasingly smaller die swell whereas in the experimental results the die swell remains almost constant. Authors believe that this is the result of morphological structure of the LCP inside the die. This means that the rheology of the LCPs can not always be simulated with a viscoelastic rheological model and morphological structure should also be considered for LCPs.



## Chapter 3

# Modeling Directionality

### 3.1 Introduction

Although amorphous and semi-crystalline polymers become isotropic in the molten state, LCPs keep their orientation even in the molten state and the interaction between crystals plays an important role in rheology of LCPs. Figure 3.1 shows the difference between amorphous, semi-crystalline and LCPs in solid and liquid states. As can be seen, LCPs keep their directionality even in the molten state.

The anisotropic properties of LCPs causes some unique behaviors that affect their processing conditions (Donald et al. [6]). Among these behaviors are low viscosity compared to conventional polymers and very small or no die swell during the extrusion. Moreover, because of the anisotropy of LCPs, the orientation of these crystals determines the properties of the final manufactured product. As an example, during the extrusion processes, crystals align themselves parallel to the direction of shear. But unlike amorphous polymers, molecules in LCPs have higher inertia which opposes the brownian motion of molecules and keeps their orientation even after exiting the die. This phenomenon makes the simulation of directionality inside the die more valuable as the directionality remains almost intact even after exiting the die. Although most of the theoretical research for modeling the rheology and orientation of LCPs is based on small molecule liquid crystals (SMLCs), the results of those studies are also applicable to polymeric liquid crystals, especially in the case of shear flows (Donald et al. [6]). The theories for the simulation of the rheology of LCPs are still evolving and researchers are trying to find more accurate theories (Han [7, 8]). The widely used constitutive equations relating the stress and strain for liquid crystals are based on the Leslie-Ericksen theory which was developed by Leslie [9] based on the work of Ericksen [10]

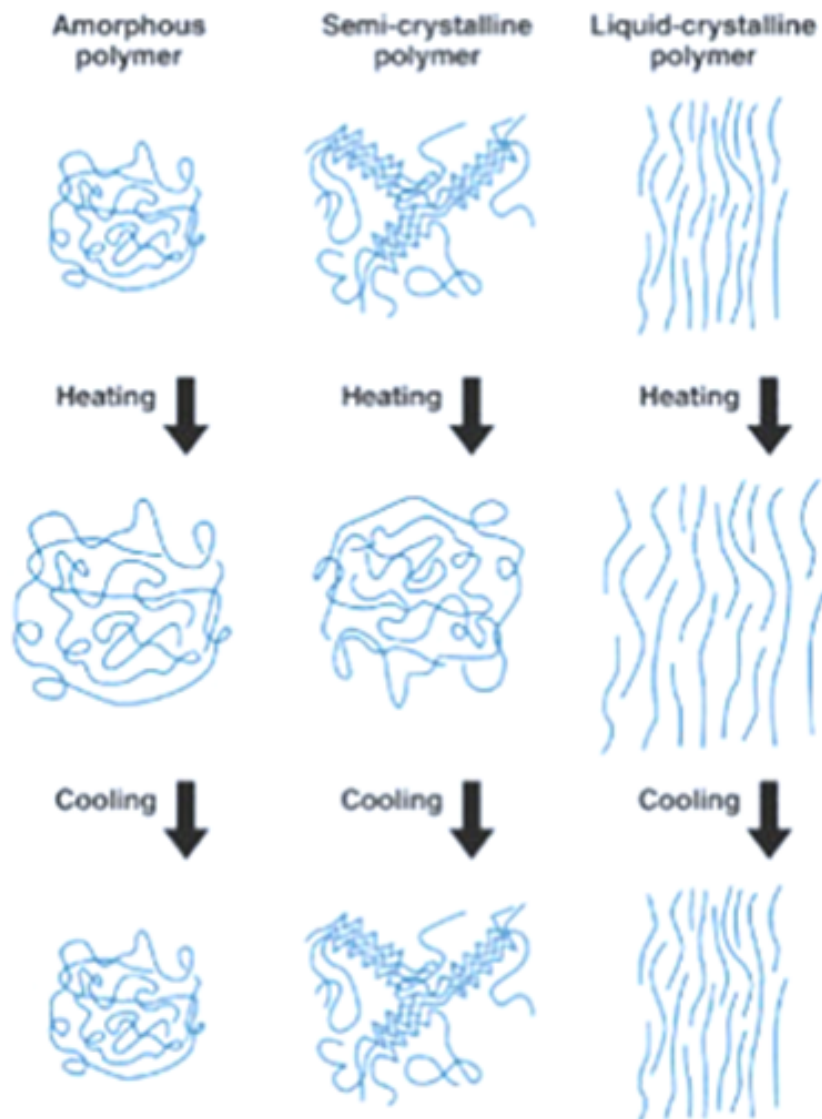


Figure 3.1: Comparison between amorphous, semicrystalline and liquid crystalline polymers

and Frank [11]. These equations consider the effect of crystals on the stresses to be continuous and defines a vector field called the director field. The equations are very complicated and only simplified variations of them, namely transversely isotropic fluid (TIF) equations have been solved numerically on simple geometries (ex. Baleo et al. [12], Chang et al. [13]). These simplified equations have some restrictions as follows:

1. Flow domain is considered to be a mono-domain without disclinations
2. Due to the high viscosity approximation, elastic stresses are considered negligible compared to viscous stresses
3. The flow is considered steady and isothermal

These assumptions pose a significant restriction to the amount of experimental data available for verification since most of the experimental data on the rheology of LCPs are based on the observation of the disclination lines between polarized glasses. The other restriction of simulating the TIF equations numerically is that the convergence of these equations is very limited, especially in complex geometries. Since the geometries of the extrusion dies are normally complex and cannot always be discretized with a structured mesh, a practical and simplified method for simulating and approximating directionality on unstructured mesh is highly desirable. The method studied in this paper has a Monte-Carlo basis and can be applied to complex geometries. The other advantage of this method is the ability to be used on a combination of structured and unstructured meshes with actual fluid parameters without convergence concerns. As a result, it can be applied in practical situations. There are some major difficulties associated with modeling the directionality on unstructured meshes. The most important one is extending the director calculations to unstructured meshes. This problem is addressed here by treating each cell locally. This means that each cell interacts with only its neighbors. Although unstructured meshes introduce some errors associated with averaging of quantities, they are the only method of discretizing the complex geometries of dies.

### 3.2 Modeling LCPs in macroscopic scale

Goldbeck-Wood et al. [22] introduced a method for modeling the directionality of crystallines on a macroscopic scale based on the Monte-Carlo approach. This hybrid method based on two separate parts. First, it considers the rheology of LCPs to be close to conventional polymers and simulates the rheology with conventional methods (ex. RANS). The second

part is the calculation of directionality assuming their directionality can be approximated by equations for small molecule liquid crystals. These two parts of the simulation are performed independently. As a result, four different calculations need to be done for the flow to be fully simulated. (a) At first the rheology of the polymer should be simulated using the existing rheological models. This step is important because the closer simulated rheology is to real flow, the more accurate the directionality modeling will be. In this step, the flow domain is meshed. Care should be taken when meshing the domain since the same cells in the mesh are going to be used later as the smallest area with aligned crystals. (b) After the rheology is simulated, velocity vectors and their gradients are used for the next step. This method assumes that each cell is small enough to have a preferred alignment of crystals throughout its volume. A vector representing the alignment of crystals is defined for each cell. These vectors are called *directors*,  $\vec{n}$ , and they are defined such that they have sense but no direction, as in crystals. This means  $\vec{n} = -\vec{n}$ . In case of liquid crystalline polymers, rigid monomers can align in a region to form a nematic phase with preferred direction parallel to  $\vec{n}$  as described by Donald et al. [6]. After extracting the results of rheological modeling, in the second step these rheological parameters are used to find the effect of rheology on directors using the evolution equation. (c) The third step is to apply the effect of Frank elastic energy on the directors. In this step, elastic stresses in the material are applied to the directors. (d) The last step is to count for the translation of crystals with the flow of the polymer. For this step a new method is developed that can be applied to each cell. The rest of this section describes how each of these steps are calculated during the simulation. The final orientation can be calculated by combining the effects of all previous steps.

Figure 3.2 shows all the steps needed for the orientation modeling in a flowchart.

### 3.3 Initial and Boundary Conditions

Three types of boundary conditions namely planar, homeotropic and tilted can be applied to the directors on the boundaries of the flow. These boundary conditions are shown in Figure 3.3. These orientations for directors can be achieved experimentally by treating the surface of the wall [29]. In this study, the directors on the boundaries are not fixed and are treated the same as the interior cells with less neighboring cells (for boundary elements  $i = 3$  and for the corner cells  $i = 2$ ). The results of Frank elastic energy is presented in the bulk of fluid far from the boundaries. In two dimensions, homeotropic and planar boundary conditions do not need any extra information to be defined. In three dimensions, the planar

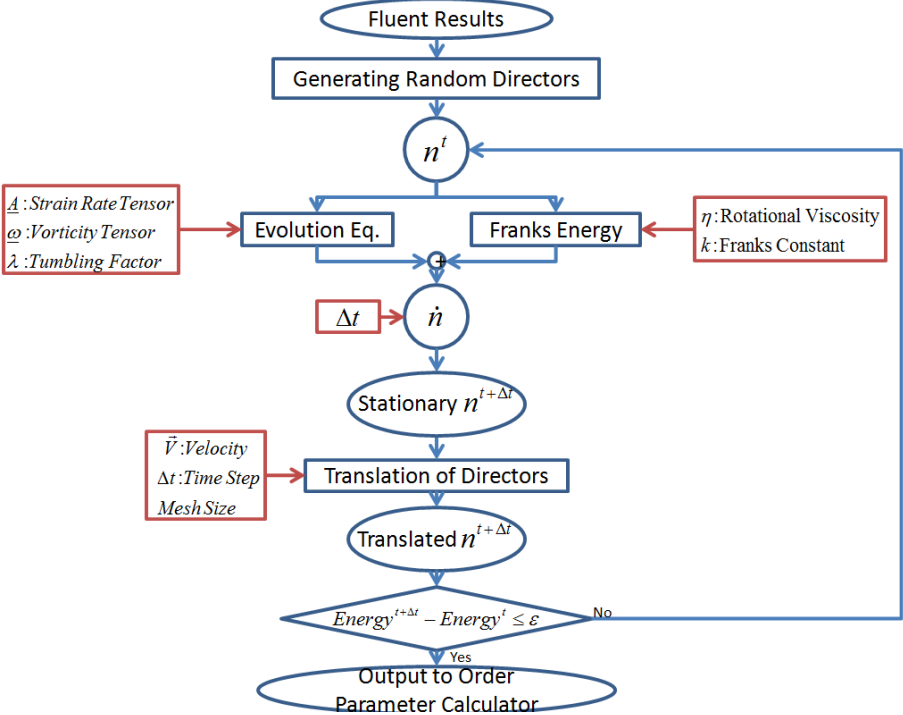


Figure 3.2: Orientation calculation flowchart

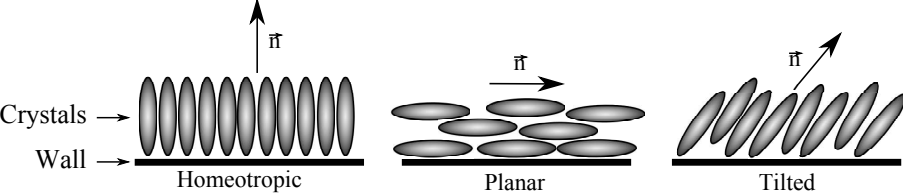


Figure 3.3: Boundary conditions on directors

boundary condition needs one angle,  $\theta$ , and the tilted boundary condition needs two angles  $\theta$  and  $\phi$  to be defined. Effect of these boundary conditions propagates inside the bulk of fluid and affects the orientation of directors inside.

The boundary conditions for the rheology are velocity inlet for the entrance and outflow for the exit. The no slip condition is applied to all other wall boundaries.

### 3.4 Franks Elastic Energy

The equation for the curvature distortion energy in liquid crystals is developed by Frank [11] and Oseen [30]:

$$F = \frac{1}{2}k_{11}(\nabla \cdot \vec{n})^2 + \frac{1}{2}k_{22}(\vec{n} \cdot \nabla \times \vec{n})^2 + \frac{1}{2}k_{33}(\vec{n} \times \nabla \times \vec{n})^2 \quad (3.1)$$

in which  $k_{11}$ ,  $k_{22}$  and  $k_{33}$  are material constants known as Frank elastic constants for splay, twist and bend respectively and  $\vec{n}$  is the director. In simple geometries, Goldbeck-Wood et al. [22] solved this equation with different values for splay, twist and bend constants. Frank elastic constants are in the order of  $10^{-10}N$  (Hobdell et al. [31]) and the relation between Frank elastic constants and molecular properties of liquid crystals are derived by Odijk [32] and Meyer et al. [33].

$$\begin{aligned} k_{11} &\approx \left(\frac{7}{8}\pi\right) \left(\frac{kT}{d}\right) \Phi \left(\frac{L}{d}\right) \\ k_{22} &\approx \left(\frac{kT}{d}\right) \Phi^{1/3} R_a^{1/3} \\ k_{33} &\approx \left(\frac{kT}{d}\right) \Phi R_a \end{aligned} \quad (3.2)$$

In equation 3.2,  $L$  is the contour length,  $k$  is the Boltzmann constant,  $R_a$  is the ratio of  $q$  (persistence length) to the diameter  $d$  of the chain,  $\Phi$  is the volume fraction which is considered to be one when dealing with thermotropic liquid crystalline polymers and  $T$  is the temperature. For LCPs, Frank elastic constants are not equal and it is shown by Hobdell and Windle [34] that more features of LCPs can be modeled when a larger value of the splay constant is considered compared to the bend and twist constants. In small molecule liquid crystals, Frank elastic constants are almost equal.

In this study, it is assumed that inside the polymer, the average alignment of crystals in each cell can be represented by a director. Moreover, for the sake of simplicity in numerical calculations, Frank elastic constants are considered to be equal. This constant here is assumed to be the average of the three Frank elastic constants derived by Hobdell et al. [31]. In this special case, equation 3.1 can be approximated by equation 3.3 and can be applied to a meshed geometry. (Lavine [35])

$$F = \frac{k}{2} \sum_{i=1}^n \sin^2(\Delta\theta_i) \quad (3.3)$$

In equation 3.3,  $k = k_{11} = k_{22} = k_{33}$  and the summation is over all the neighboring cells which have a common surface with the central cell.  $\Delta\theta_i$  is the difference between the angle of the central cell and its  $i$ -th neighbor. The torque applied to the central cell can be calculated by taking the derivative of the distortion energy with the angle  $\Delta\theta_i$ .

$$\vec{\Gamma} = \frac{dF}{d(\Delta\theta_i)} = k \sum_{i=1}^n \sin(\Delta\theta_i) \cos(\Delta\theta_i) \quad (3.4)$$

The torque,  $\vec{\Gamma}$ , is the result of distortion in the liquid crystals. Since for the two dimensional case the direction of the torque vector is always perpendicular to the plane of flow, it is possible to do the algebraic summation over all the neighbors and find the resulting torque applied to the central cell. In contrast, in the three dimensional case the effect of each neighbor on the central cell should be calculated separately and added like vectors. In three dimensional geometries the effect of each neighbor on the central cell will result in a torque  $\vec{\Gamma}_i$  which is normal to the plane that passes through the two directors having the same origin. As a result, the total torque  $\vec{\Gamma}$  applied to the central cell is the vector summation of these individual vectors ( $\vec{\Gamma}_i$ ).

After finding the resultant torque applied to the central cell, one can find the rate of change of direction of the director as

$$\dot{\vec{n}} = \frac{\vec{\Gamma}}{\zeta_r} \quad (3.5)$$

in equation 3.5  $\zeta_r$  is the rotational viscosity which is based on Doi and Edwards [36] for thermotropic liquid crystalline polymers ( $\Phi = 1$ ) and can be approximated as

$$\zeta_r = \beta\eta R_a^2 \quad (3.6)$$

in which  $\beta$  is an empirical constant in the order of  $10^{-4}$  (Larson [37]). For a specific Vectra-A material with a typical viscosity of  $\eta = 10^{-3} Pa.s$ , rotational viscosity is about  $\zeta_r = 10 Pa.s$  (Goldbeck-Wood et al. [22]). Figure 3.4 shows the flowchart of the process of applying the Franks elastic energy to the random generated directors.

### 3.5 Evolution Equation

Evolution equation describes the effect of the rheology on directors. This equation considers the directors as if they were pinned on their center of mass and can only spin without translation as described in Donald et al. [6]

$$\dot{\vec{n}} = \vec{n} \cdot \underline{\omega} + \lambda (\vec{n} \cdot \underline{A} - (\vec{n} \cdot \underline{A} \cdot \vec{n}) \vec{n}) \quad (3.7)$$

The first term on the right hand side of equation 3.7 accounts for the effect of the vorticity tensor,  $\underline{\omega}$ , and the second term is to calculate the effect of the strain rate tensor,  $\underline{A}$ , on the

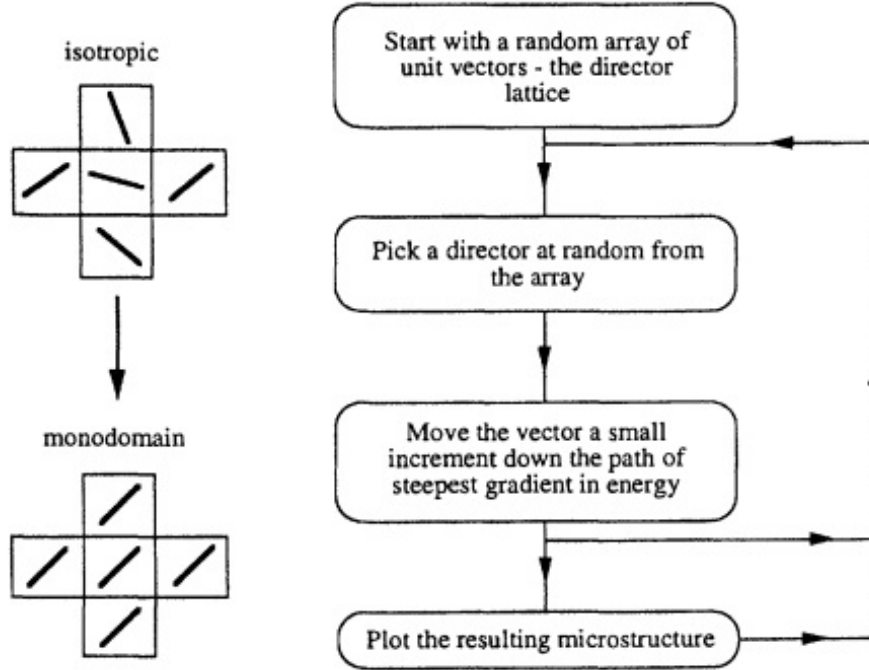


Figure 3.4: Franks elastic energy minimization flowchart

director  $\vec{n}$ .  $\underline{\omega}$  and  $\underline{A}$  are antisymmetric and symmetric parts of the velocity gradient tensor.  $\lambda$  is the tumbling factor defined in Ericksen [10] which shows the relative importance of the effect of the vorticity tensor to the strain rate tensor. A physical interpretation of the tumbling parameter,  $\lambda$ , can be sought if ellipsoids are considered instead of directors in the field. With this assumption, for the case of  $\lambda \leq 1$ ,  $\lambda$  can be related to the molecular characteristics of the LCP chains.

$$\lambda = \frac{R_a^2 - 1}{R_a^2 + 1} \quad (3.8)$$

In equation 3.8,  $R_a$  is the ratio of the length of the major axis of ellipsoids to the length of the minor axis which in the case of the liquid crystalline molecules will be the ratio of the persistence length over the diameter of the chains. Three different regions for  $\lambda$  are normally considered.  $\lambda < 1$  is associated with the case that the effect of vorticity tensor dominates and tumbling of directors occurs. In this case, ellipsoids are close to spheres.  $\lambda = 1$  is the pseudo-affine case. Based on discussions of Lavine and Windle [23], the value  $\lambda = 1$  is more suitable for thermotropic liquid crystalline polymers.  $\lambda > 1$  which can not be interpreted using equation 3.8, is the case which the effect of the strain rate tensor is dominant and gives a stable alignment with a positive value of Leslie angle (Larson [38]). Solving the vector



equation 3.7 gives the rate of change of the director  $n$  with time in three principal directions namely  $\frac{dn_x}{dt}$ ,  $\frac{dn_y}{dt}$  and  $\frac{dn_z}{dt}$ . By having the time steps of the simulation,  $\Delta t$ , changes in the director,  $\vec{dn}$ , and its three components  $dn_x$ ,  $dn_y$  and  $dn_z$  can be calculated for each cell.

### 3.6 Translation of Directors

Evolution and Frank elastic energy equations assume directors to be fixed in place and can only spin around their center of mass. However, in real flows, directors will be translated with the bulk of material which needs to be accounted for in the numerical simulation. In the work of previous researchers like Lavine and Windle [23], structured meshes were considered and the Poiseuille flow was considered parallel to the x-direction. This method can not take into account the real flow phenomena like vortices and separations. Since in this study different element types and orientations are considered, a new method for simulating the translation of directors is developed.

The translation of directors is calculated from one element to the neighboring ones based on the direction and velocity of the flow for each element. Figure 3.5 shows a simple case in which a quadrilateral central cell,  $c$ , is surrounded by four other cells  $i = 1 \dots 4$ . The method used here assumes that the velocity  $\vec{V}$  transfers bulk of fluid to cell  $i$  only if  $\vec{V} \cdot \vec{A}_i > 0$ . So for the mesh shown in Figure 3.5, bulk of fluid is transferred from central cell to the 1st ( $i = 1$ ) and 2nd ( $i = 2$ ) neighbor due to the acute angle between  $\vec{V}$  and  $\vec{A}_i$ . The important step in this simulation is that the changes needed for all the cells should be calculated and added together before applying it to the neighboring cell because of the fact that each cell may receive crystals from more than one neighbor. It is also important to consider the value of the velocity in calculating the transfer of fluid. Here the velocity of fluid in each cell is compared to the distance between cell centers to determine how each cell affects its neighbors. This method can be applied to cells regardless of their number of faces in two and three dimensions. An important consideration here is that the average time step of the simulation should be chosen such that the translation of directors is comparable to the average distance between cell centroids of neighboring cells.

### 3.7 Order Parameter

Characterization of the orientation in liquid crystals is usually done using order parameters. Order parameters are defined for different phases of liquid crystals and play an important role in describing the phase transitions between mesophases.

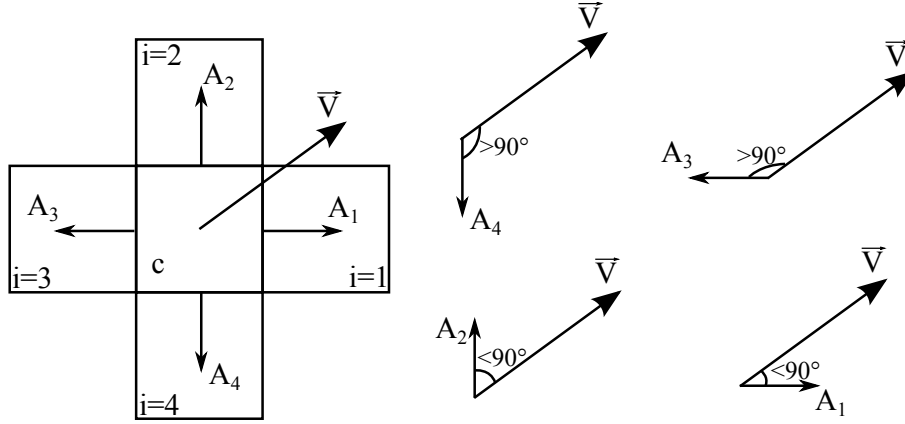


Figure 3.5: Translation of directors

Nematic liquid crystals can be represented as a pool of rigid rods. If  $\vec{a}$  represents the long axis of these rods, then the ordering in the system considered can be characterized by the local mean quadric combination of the components of vectors in a local coordinate system. [39] [40]

$$Q_{ij} = \frac{1}{N} \sum_N \left( a_i a_j - \frac{1}{3} \delta_{ij} \right) \quad (3.9)$$

where  $N$  is the number of crystals in a small but macroscopic volume. ( $\delta_{ij} = 1$  if  $i = j$  and  $\delta_{ij} = 0$  if  $i \neq j$ ).  $Q_{ij}$  is the tensor order parameter which is a polar symmetric tensor of the second rank. If the nematic crystals are distributed randomly then  $Q_{ij} = 0$ . In the case of alignment of crystals in a preferred direction, the angles between vectors  $\vec{a}$  decreases and  $Q_{ij}$  increases. Since the director  $\vec{n}$  in a nematic is the average orientation of crystals  $\vec{a}$ , the tensor order parameter can be easily expressed through the director components

$$Q_{ij} = Q \left( n_i n_j - \frac{1}{3} \delta_{ij} \right) \quad (3.10)$$

where scalar  $Q$  represents the proportion of molecules that are oriented in a given direction. Order parameter tensor  $Q_{ij}$  has some important properties.

- Since  $\delta_{ij} = \delta_{ji}$  and  $n_i n_j = n_j n_i$ , order parameter tensor is a symmetric tensor.
- Due to the fact that  $\vec{n}$  is a unit vector, trace of  $Q_{ij}$  is zero.

$$Tr Q_{ij} = \sum_{i=(1,2,3)} Q_{ii} = \frac{1}{N} \sum \left[ (n_1)^2 + (n_2)^2 + (n_3)^2 - \frac{1}{3} 3 \right] = 1 - 1 = 0 \quad (3.11)$$

- Two previous properties of  $Q_{ij}$  reduce the number of independent components from 9 to 5
- In the isotropic phase  $Q_{ij}^{iso} = 0$

It is possible to prove this by transforming the components of the director to spherical coordinate system.

$$\begin{aligned} n_1 &= \sin\theta\cos\phi \\ n_2 &= \sin\theta\sin\phi \\ n_3 &= \cos\theta \end{aligned} \quad (3.12)$$

Then

$$Q_{ij} = \int_0^{2\pi} d\phi \int_0^\pi \sin\theta d\theta f(\theta, \phi) \left( n_i n_j - \frac{1}{3} \delta_{ij} \right) \quad (3.13)$$

where  $f(\theta, \phi)$  is the probability function which represents the probability to find a molecule with the orientation given by angles  $\theta$  and  $\phi$ . Clearly in the isotropic phase  $f$  is a constant. The probability in isotropic case,  $f^{iso}$ , should be normalized with

$$2 \int_0^{2\pi} d\phi \int_0^{\pi/2} \sin\theta d\theta = 4\pi \quad (3.14)$$

This means that  $f^{iso}(\theta, \phi) = \frac{1}{4\pi}$ . It should be noted that the integration over  $\theta$  is from 0 to  $\pi/2$  but the integration is multiplied by two to compensate for that. This is due to the fact that  $\vec{n} = -\vec{n}$ . It is easy to see that in the isotropic phase  $Q_{12} = Q_{23} = Q_{13}$  because of the integration over  $\phi$  (Periodic function integrated over its full period). For the diagonal components  $Q_{33}$  can be written as:

$$\begin{aligned} Q_{33} &= 2 \int_0^{2\pi} d\phi \int_0^{\pi/2} \sin\theta d\theta f(\theta, \phi) \left( \cos^2\theta - \frac{1}{3} \right) = \\ &4\pi f_{iso} \int_0^1 \left( \cos^2\theta - \frac{1}{3} \right) d(\cos\theta) = \frac{2}{3}\pi(x^3 - x)_0^1 = 0 \end{aligned} \quad (3.15)$$

With a similar method it can be shown that other diagonal components are also zero.

- In a nematic case where the crystals are aligned in the direction of  $\vec{n}$  (probate geometry)

$$Q^{probate} = \begin{pmatrix} -1/3 & 0 & 0 \\ 0 & -1/3 & 0 \\ 0 & 0 & 2/3 \end{pmatrix} \quad (3.16)$$

Proof for  $Q_{33}$  is

$$Q_{33} = n_3 n_3 - 1/3 = 1 - 1/3 = 2/3 \quad (3.17)$$

- In a perfectly aligned oblate arrangement where  $n_3 = 0$

$$Q^{oblate} = \begin{pmatrix} 1/6 & 0 & 0 \\ 0 & 1/6 & 0 \\ 0 & 0 & -1/3 \end{pmatrix} \quad (3.18)$$

These properties show that  $Q_{ij}$  is a very good candidate for the order parameter of nematic liquid crystals. It is equal to zero for the isotropic alignment, it is sensitive to the direction of the average orientation of the molecules as well as to the molecular distribution of the crystals around their average orientation. Order parameter tensor is often used to describe complicated situations in which the order parameter is a function of coordinates. Other than that, a scalar order parameter  $Q$  is sufficient.

$Q$  can be expressed in terms of the distribution function  $f$ . The distribution function determines the mutual orientation of the crystals. If the long axis of a crystal is considered to be in the  $z$  direction, then the orientation of the second crystal can be determined in a cartesian coordinate system as:

$$a_1 = \sin\theta\cos\phi, \quad a_2 = \sin\theta\sin\phi, \quad a_3 = \cos\theta \quad (3.19)$$

The value  $f(\theta, \phi)d\Omega$  is the probability that the long axis of the second rod placed inside a little angle  $d\Omega = \sin\theta d\theta d\phi$  formed around the direction  $(\theta, \phi)$ . For nematic liquid crystals the distribution function is independent of  $\phi$  and  $f(\theta) = f(\pi - \theta)$  (non-polar crystals). For nematic liquid crystals  $Q$  in equation 3.10 can be written as

$$Q = \int f(\theta) \frac{1}{2} (3\cos^2\theta - 1) d\Omega = \frac{1}{2} \langle (3\cos^2\theta - 1) \rangle = \langle P_2(\cos\theta) \rangle \quad (3.20)$$

where  $P_2$  is the Legendre polynomials of the second order. It can be seen from equation 3.20 that if  $f(\theta)$  has a sharp peak for  $\theta = 0$  and  $\theta = \pi$  (all the crystals are aligned with the director), then  $\cos\theta = \pm 1$  and  $Q = 1$ . On the other hand, if the peak is at  $\theta = \pi/2$  (crystals oriented perpendicular to the director), then  $Q = -\frac{1}{2}$ . And the third extreme case the orientation of crystals is completely random which makes  $f$  independent of  $Q$ , then  $\langle \cos^2\theta \rangle = \frac{1}{3}$  and  $Q = 0$ . Any case in between results in a value of  $Q$  between 0 and 1.

$Q$  characterizes the molecular ordering in nematic liquid crystals and is called the scalar order parameter.[41]

### 3.8 Numerical Method

As described above, the directionality simulation used in this study has been implemented with user defined functions (UDFs) in Fluent. UDFs are powerful tools available in *ANSYS® FLUENT®* to increase the capabilities of *ANSYS® FLUENT®*. Since all the code written for UDFs are in C++ language, it is possible to utilize all the capabilities of C++ programming and functions for the calculations. There is no readily available tool in *ANSYS® FLUENT®* to simulate the directionality of fluid; as a result, UDFs are used for this simulation. Three independent functions are used to calculate the effects of Frank elastic equation, evolution equation and translation of directors. Vorticity tensor,  $\underline{\omega}$ , and strain rate tensor,  $\underline{A}$ , should be built by extracting the velocity gradients in different directions from the solver. Six user defined memories (UDMs) are needed in 2D to complete the calculation. UDMs are defined to store a variable for each cell in *ANSYS® FLUENT®*. It is possible to store the components of a director for each cell using UDMs.

To calculate the order parameter, a function called *OrderParameter()* was added to the user defined function. This function uses the equation 3.20 and integrates over a range of angles from 0 to  $\pi$ .

Four different sets of results are presented here to show the ability of the proposed method in simulating the orientation of crystals. The first three sets of results are presented to verify the isolated effects of Frank elastic energy, evolution equation and translation of directors, respectively. The last set of results is the final solution which is the combination of the first three calculations.

The effect of Frank elastic energy function on randomly generated directors are shown in Figure 3.6. In this figure, there is no additional influence on the directors except for their interaction with each other. This result shows what happens when liquid crystals are left to annihilate in a quiescent condition. This figure shows a close-up of the directors. Two of the disclinations that exist in the structured mesh are shown with the strength of  $s = 1/2$  and  $s = -1/2$ . Disclinations are less obvious on the unstructured mesh because of the random placement of the directors in the domain. If the annihilation continues, all the disclinations disappear and domain turns into a mono-domain[42].

The result of applying the effect of shear to a randomly generated director field on (a) structured and (b) unstructured mesh are shown in Figure 3.7. As can be seen, the effect of shear penetrated inside the fluid. Flow entering the channel from left has a uniform velocity profile and as it moves through the channel, the boundary layer develops and makes the directors aligned to the direction of flow. In both cases of structured and unstructured

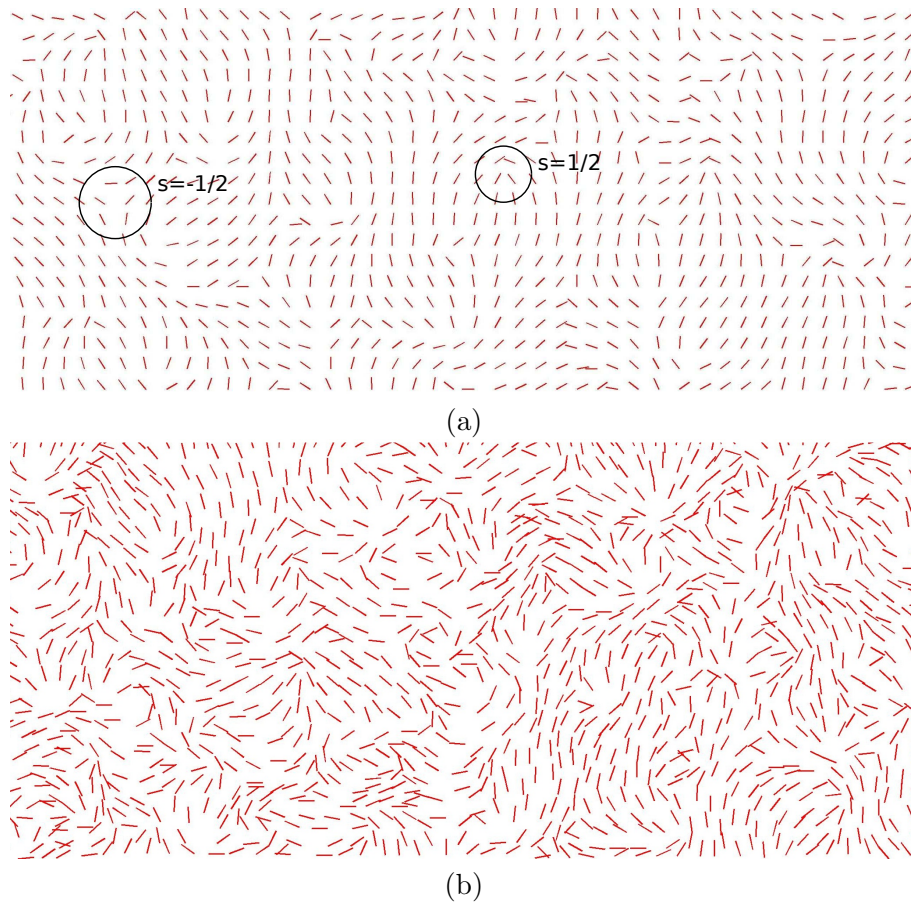


Figure 3.6: Effect of Frank elastic energy on directors in (a) structured mesh and (b) unstructured mesh

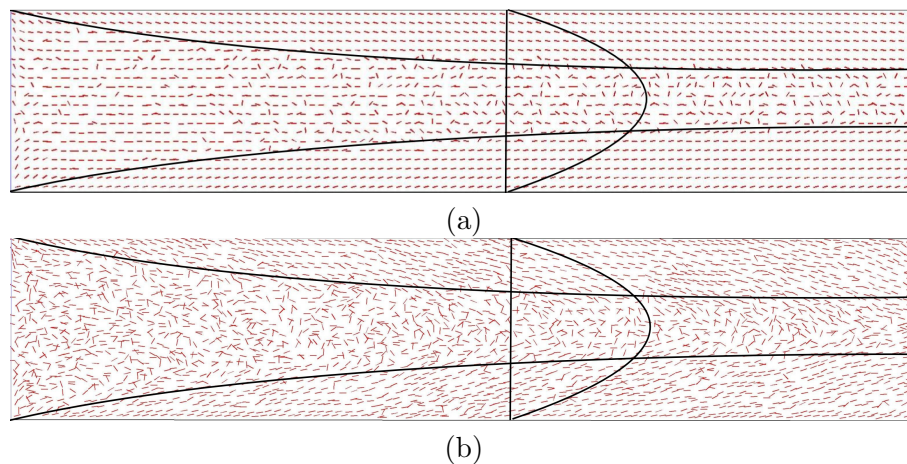


Figure 3.7: Effect of shear on directors in (a) structured mesh and (b) unstructured mesh

meshes, the initial random directors remain undisturbed in the middle of the channel where minimum shear occurs. An approximate line is drawn to show the extent of directors affected by shear forces.

Figures 3.8 and 3.9 show how the described method for translating the directors affects the director field. The meshes considered for this calculation are the same meshes used in previous calculations and the initial condition is such that  $1/3$  of the directors are positioned vertically and in the rest of the domain directors are positioned horizontally. The mass flow rate is lowered in this calculation so that all the stages of translation of directors can be visualized. As can be seen in Figure 3.9, the lower velocity of fluid in the boundary layer is preventing the flow to transfer the directors to the nearby cells. On the other hand, the higher velocity of the bulk of fluid in the center of the channel transfers the directors to the downstream cells completely. The important consideration in this step is to choose the mass flow rate and time step such that the average translation of directors in each time step  $\Delta t$  will be almost the same distance as mesh sizes.

The last set of results shown in Figure 3.10 are the combination of all previous steps with matching time steps. Frank elastic constant is  $k = 10^{-10}$  and  $\Delta t = 10^{-4}s$ . The boundary condition for the inlet is velocity inlet with  $V = 5m/s$ . The tumbling parameter in these simulations is considered to be  $\lambda = 1.05$  to avoid instabilities in the solution but unlike direct simulation of the Leslie-Ericksen equation,  $\lambda = 1$  can also be used without any divergence in the solution. The boundary condition for the directors at the inlet of the channel has to reflect the actual physical condition for the inlet. In this simulation, it is assumed that the flow entering the channel has a random orientation. As a result, at

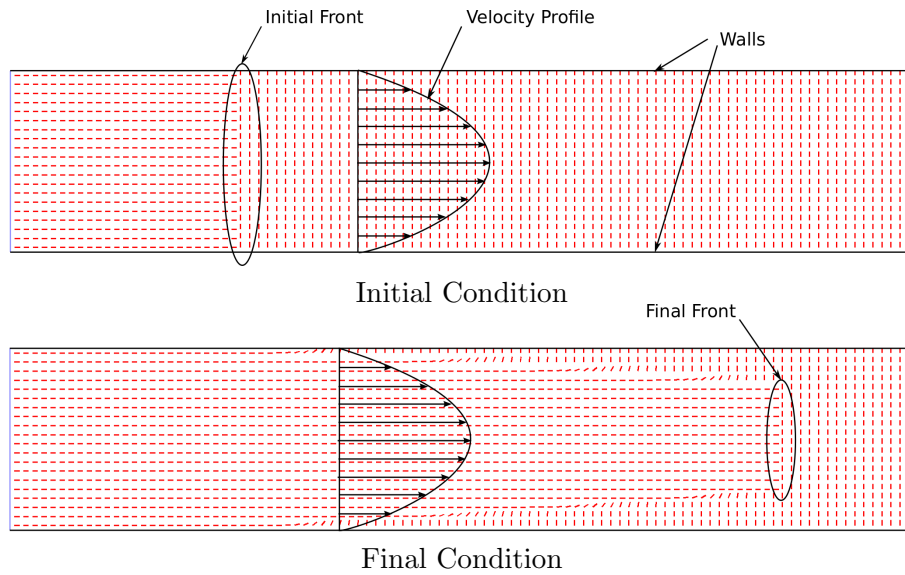


Figure 3.8: Effect of translation of directors with bulk of fluid on structured mesh

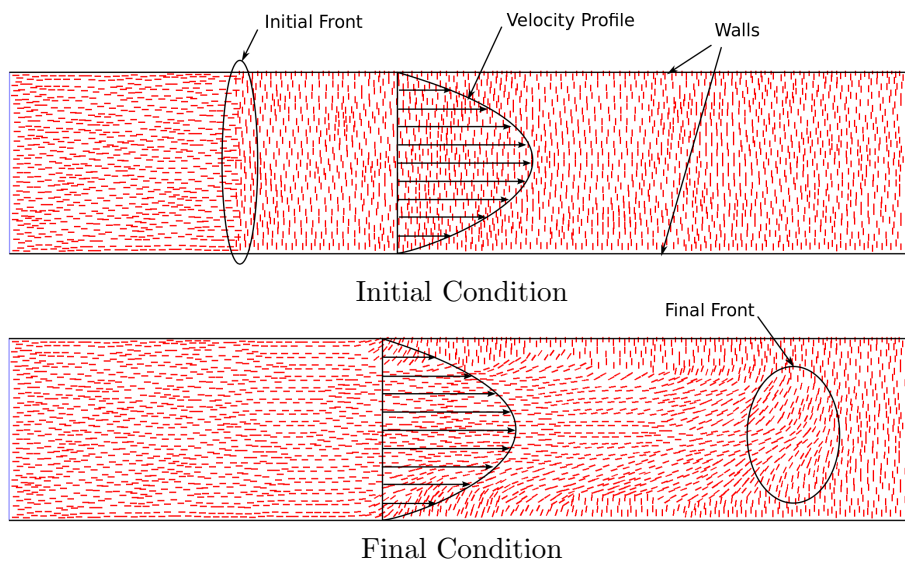


Figure 3.9: Effect of translation of directors with bulk of fluid on unstructured mesh



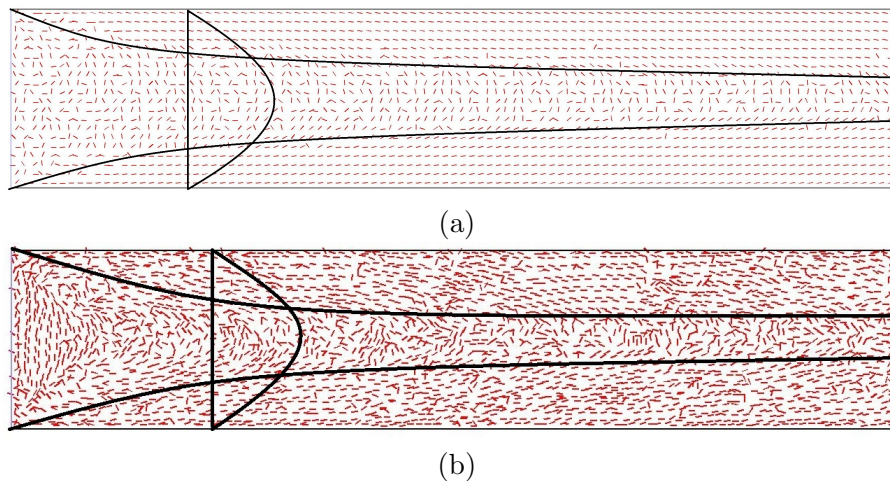


Figure 3.10: Final simulation of orientation (combination of all three steps) for (a) structured mesh (b) unstructured mesh

the inlet boundary of the channel, the orientation of directors are randomly generated for each time step. Figure 3.10 shows aligned directors along shear and random directors in the middle of the channel.

The differences between the results shown for structured and unstructured meshes are probably due to the fact that the calculation of averages in structured and unstructured meshes are affecting the alignment of crystals. One other possibility for the discrepancy might be due to the fact that the translation of directors between cells are affected by the shape and arrangement of cells. Increasing the accuracy of the calculation of averages and taking the effect of partial translation of fluid between cells can improve the results.

The following results show how changing each of the material or flow properties affect the order parameter. As mentioned before, order parameter is an important characteristic of the LCPs that quantifies the order in these polymers. The geometry of the flow is the same as 3.10. Here the parameters that are changing are the inlet velocity, power-law constant, and the temperature difference between the inlet and the walls.

Figure 3.11 shows how increasing the velocity at the inlet of the channel affects the order parameter. As can be seen the order parameter increases after applying the effect of evolution equation. This means that increasing the velocity increases the shear and turns the directors to be aligned in the direction of the shear. The same effect can be seen when the effects of all three components are combined. After velocity increases beyond  $3 \text{ m/s}$ , the order parameter for the combination reduces although the order parameter still increases for the evolution. This can be the result of the translation of directors with the bulk of fluid.

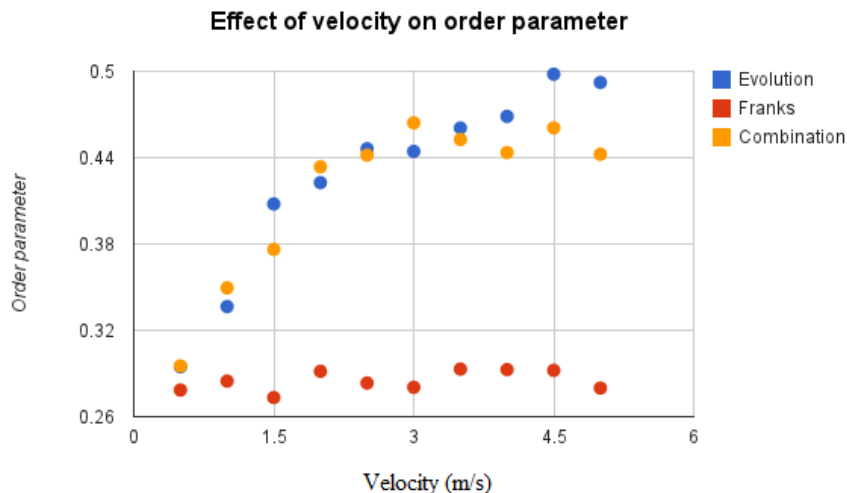


Figure 3.11: Effect of increasing the inlet velocity on the order parameter

When a director is transferred inside the flow domain from the entrance, UDF replaces it with a newly random generated director. These randomly generated directors can be translated inside the flow domain and disturb the alignment of the crystals at higher speed. Increasing the inlet velocity by itself does not affect the Franks elastic constants since the mechanism for the Franks elastic energy is acting based of the orientational elasticity.

In Figure 3.12 the order parameter is calculated after changing the power-law index in the viscosity equation (Eq. 2.1). The order parameter decreases from 0.1 to 0.3 and then increases. The increase from 0.3 to 1 can be interpreted as the effect of bigger boundary layer which makes more of the directors to be affected by the shear in the boundary later. The decrease from 0.1 to 0.3 maybe due to the decrease in the effect of the translation of director which let the boundary layer to affect the directors more.

To investigate the effect of temperature on the rheology and directionality, the simulation is done with finite temperature difference between the inlet fluid and the walls. The temperature difference between the inlet and the walls causes the viscosity of the polymer to increase closer to the (cold) walls and increases the thickness of the boundary layer. Figure 3.13 shows how increasing the temperature difference causes a slight increase in the order parameter. In this simulation, the inlet velocity is  $1\text{ m/s}$  and the diffusivity of the polymer is increased to make the effect of the temperature difference more observable. Temperature difference will not have an effect of the Franks elastic constants in our method.

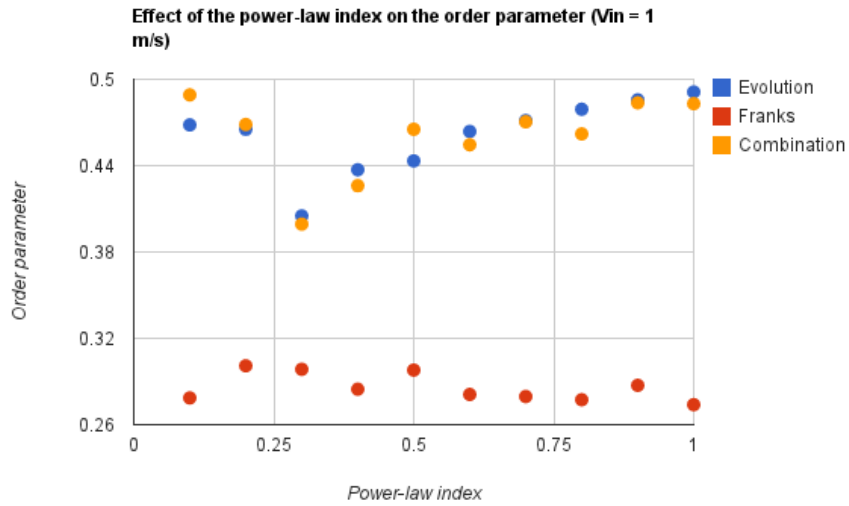


Figure 3.12: Effect of the power-law index on the order parameter

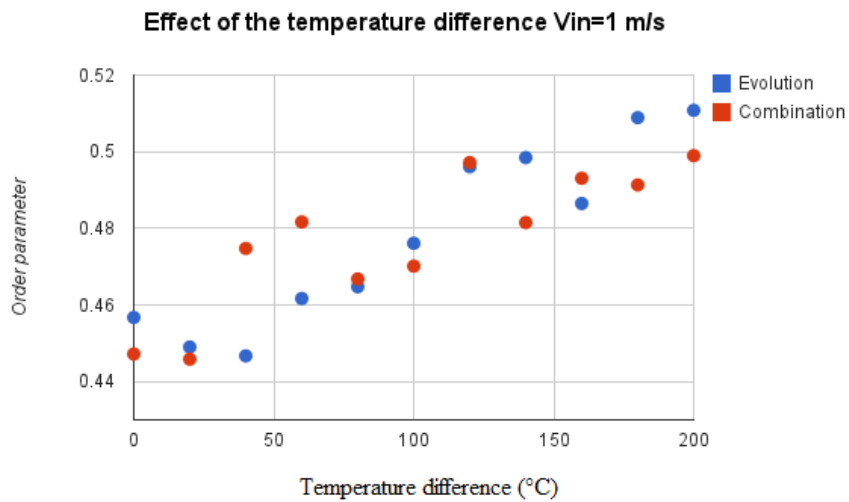


Figure 3.13: Effect of the temperature difference between inlet and the walls on the order parameter

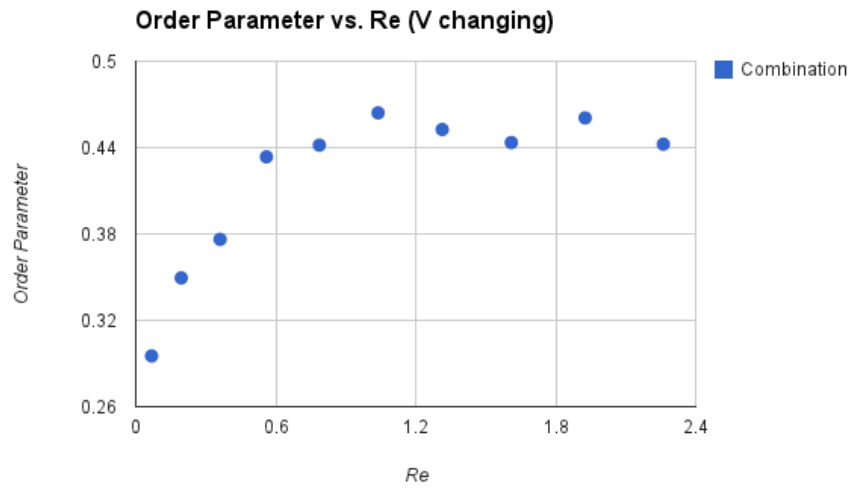


Figure 3.14: Change of order parameter vs.  $Re$  when average velocity  $\bar{V}$  is changing

It is possible to calculate the Reynolds number of the flow for power-law fluids as described in Appendix B. In power-law fluids,  $Re$  depends on both the average velocity of the flow and the power-law index. For the case of changing the velocity of the flow, the order parameter is plotted against  $Re$  number. (Figure 3.14) The trend for the dependence of  $Re$  is the same as the trend for velocity.

## Chapter 4

# Conclusion and Future Work

Numerical simulations and experimental measurements done in this thesis were mainly to achieve a more practical method to model the directionality of liquid crystalline polymers during their processing. For this purpose A method for simulating the directionality in finite volume software ANSYS<sup>®</sup> FLUENT<sup>®</sup> based on the Monte-Carlo method is studied and the result of a simple 2D case is presented. The effects of each component of directionality modeling is studied and the physical interpretation of each of the components is discussed. This method is developed to simulate the orientation of crystals in LCP materials where a rough estimation of directionality is needed.

The main advantage of this method compared to the direct solution of the Leslie-Ericksen equations is its stability for complex geometries. Geometries of the extrusion dies for processing LCPs are complex and can not be meshed using structured meshes. Other advantages of this method are (a) the ability to use experimental rheological results to simulate the rheology, (b) ease of convergence (c) potential for 3D modeling in complex geometries and (d) the potential to model the disclinations in LCPs. The major difficulty arises when dealing with unstructured meshes. In this study, the translation of directors are dealt with locally which makes it possible to solve for any shape and combination of meshes. This method works best if the transition of the size of the cells happens very gradually and the aspect ratio of the cells are almost one. For better accuracy of the high aspect ratio unstructured meshed, a more robust averaging algorithm is needed. By comparing different components of directionality modeling, it can be concluded that the effect of Frank elastic energy on the flow is negligible compared to the effect of shear and the translation of directors with the bulk of fluid.

The presented results show promising correlation to physical phenomena governing the

rheology and orientation of LCPs in shear flows. These phenomenon are alignment of crystals along the direction of shear and annihilation of defect structure in quiescent condition due to Franks equation.

Some experimental measurements were also carried out to develop a better understanding of the rheology of LCPs. Two capillary dies based on the oscillatory rheometry results was designed for a liquid crystalline polymer material. The die swell of a LCP and an amorphous polymer (polypropylene) was measured optically about 5mm after exiting the die.

The measurements show that the diameter of the extrudate is almost the same as the diameter of the die for the case of the LCP and no die swell or negative die swell exists. On the other hand, the die swell of the polypropylene is significantly larger which is known to be due to the existance of elasticity and normal stress differences. An attempt to model the rheological properties of the LCP based on Phan Thien-Tanner (PTT) viscoelastic model was made using ANSYS<sup>®</sup> POLYFLOW<sup>®</sup> in a 2D axi-symmetric geometry. PTT model is one of the most realistic differential viscoelastic models that can capture the shear thinning behavior of the LCP. It also exhibits non quadratic normal stress differences at high shear rates. The material parameters of the PTT model and the curve fitting parameters in ANSYS<sup>®</sup> POLYMAT<sup>®</sup> were changed to achieve the experimental results for the die swell.

The results show that although it is possible to get the desired results for the first volume flow rate/shear rate, increasing the volume flow rate changes the rearrangement of the velocity field at the lip of the die and causes excessive decrease in the diameter of the extrudate. Based on these results it appears that the consideration of the directionality and crystallization of LCP is important in successful modeling of their die swell.

To improve the overall modeling of the directionality it is important to improve the rheological and directionality modeling at the same time. It is possible to perform more experimental measurements for the extensional viscosity, second normal stress differences, ... to be able to include more sophisticated fluid models in the rheological modeling. Extending the directionality modeling to 3D is also very important since in reality disclinations appear in curved lines. Improving the method of translation of directors for the case of unstructured meshes will also improve the results.

# Bibliography

- [1] Dane C Thompson, Olivier Tantot, Hubert Jallageas, George E Ponchak, Manos M Tentzeris, and John Papapolymerou. Characterization of liquid crystal polymer (lcp) material and transmission lines on lcp substrates from 30 to 110 ghz. *Microwave Theory and Techniques, IEEE Transactions on*, 52(4):1343–1352, 2004.
- [2] W Chinsirikul, TC Hsu, and IR Harrison. Liquid crystalline polymer (lcp) reinforced polyethylene blend blown film: Effects of counter-rotating die on fiber orientation and film properties. *Polymer Engineering & Science*, 36(22):2708–2717, 1996.
- [3] Richard W Lusignea. Orientation of lcp blown film with rotating dies. *Polymer Engineering & Science*, 39(12):2326–2334, 2004.
- [4] D Boles, M Cakmak, and B Yalcin. A multi-stream flow technique to obtain isotropic texture in extruded thermotropic liquid crystalline polymer melts. *Polymer Engineering & Science*, 49(1):73–80, 2008.
- [5] Shu-F Fu and Xuan-zhang Wang. Effects of magnetic field on shear stress of nematic liquid crystalline polymer under simple shear flow. *Journal of Polymer Science Part B: Polymer Physics*, 48(17):1919–1926, 2010.
- [6] A.M. Donald, A.H. Windle, and S. Hanna. *Liquid crystalline polymers*. Cambridge Univ Pr, 2006.
- [7] Shifang Han. New conception in continuum theory of constitutive equation for anisotropic crystalline polymer liquids. *Natural Science*, 2(9):948–958, 2010.
- [8] Shifang Han. A new constitutive theory for extrusion-extensional flow of anisotropic liquid crystalline polymer fluid. *Natural Science*, 3(4):307–318, 2011.
- [9] F. M. Leslie. Some constitutive equations for liquid crystals. *Archive for Rational Mechanics and Analysis*, 28:265–283, 1968. ISSN 0003-9527. 10.1007/BF00251810.

- [10] J. Ericksen. Transversely isotropic fluids. *Colloid & Polymer Science*, 173:117–122, 1960. ISSN 0303-402X. 10.1007/BF01502416.
- [11] F. C. Frank. I. liquid crystals. on the theory of liquid crystals. *Discuss. Faraday Soc.*, 25:19–28, 1958.
- [12] J.N. Baleo, M. Vincent, P. Navard, and Y. Demay. Finite element simulation of flow and director orientation of viscous anisotropic fluids in complex 2d geometries. *Journal of rheology*, 36(4):663–701, 1992.
- [13] R.Y. Chang, F.C. Shiao, and W.L. Yang. Simulation of director orientation of liquid crystalline polymers in 2-d flows. *Journal of non-newtonian fluid mechanics*, 55(1): 1–20, 1994.
- [14] GA Hill and CL Chenier. Die swell experiments for newtonian fluids. *The Canadian Journal of Chemical Engineering*, 62(1):40–45, 1984.
- [15] G. Russo and T. N. Phillips. Numerical simulation of steady planar die swell for a newtonian fluid using the spectral element method. *Computers & Fluids*, 39(5):780–792, 2010.
- [16] Thomas Schweizer. Measurement of the first and second normal stress differences in a polystyrene melt with a cone and partitioned plate tool. *Rheologica acta*, 41(4): 337–344, 2002.
- [17] J Meissner, RW Garbella, and J Hostettler. Measuring normal stress differences in polymer melt shear flow. *Journal of Rheology*, 33(6):843–864, 1989.
- [18] Richard Lusignea, Kent Blizard, and Leslie Rubin. High performance processing for high performance polymers. *MRS Online Proceedings Library*, 305, 1993.
- [19] B. Farrell and M. St Lawrence. The processing of liquid crystalline polymer printed circuits. In *Electronic Components and Technology Conference, 2002. Proceedings. 52nd*, pages 667–671, 2002.
- [20] R. Ahmed, R.F. Liang, and M.R. Mackley. The experimental observation and numerical prediction of planar entry flow and die swell for molten polyethylenes. *Journal of Non-Newtonian Fluid Mechanics*, 59(23):129–153, September 1995. ISSN 0377-0257.
- [21] FLUENT. Version 13, ansys inc, 2010.



- [22] G. Goldbeck-Wood, P. Coulter, J. R. Hobdell, M. S. Lavine, K. Yonetake, and A. H. Windle. Modelling of liquid crystal polymers at different length scales. *Molecular Simulation*, 21(2-3):143–160, 1998.
- [23] Marc S. Lavine and Alan H. Windle. Computational modelling of declination loops under shear flow. *Macromolecular Symposia*, 124(1):35–47, 1997. ISSN 1521-3900.
- [24] A. Ahmadzadegan, A. Saigal, and M. A. Zimmerman. Investigating the rheology of lcp through different die geometries. In *Proceedings of 2012 TMS Annual Meeting, Characterization of Minerals, Metals, and Materials*, pages 349–356, Walt Disney World, FL, USA, March 2012.
- [25] ANSYS POLYFLOW. Ver. 12.1 - user’s manual, September 2009.
- [26] N. P. Thien and R.I. Tanner. A new constitutive equation derived from network theory. *Journal of Non-Newtonian Fluid Mechanics*, 2(4):353–365, 1977.
- [27] ISO 11443. Plastics, Determination of the fluidity of plastics using capillary and slit-die rheometers, 2000.
- [28] Jan-Chan Huang and Zhenghong Tao. Melt fracture, melt viscosities, and die swell of polypropylene resin in capillary flow. *Journal of applied polymer science*, 87(10):1587–1594, 2003.
- [29] Linda T. Creagh and Allan R. Kmetz. Mechanism of surface alignment in nematic liquid crystals. *Molecular Crystals and Liquid Crystals*, 24(1-2):59–68, 1973.
- [30] CW Oseen. The theory of liquid crystals. *Trans. Faraday Soc.*, 29(140):883–899, 1933.
- [31] J.R. Hobdell, M.S. Lavine, and A.H. Windle. Hierarchical approach to modelling of liquid-crystalline polymers. *Journal of Computer-Aided Materials Design*, 3(1):359–368, 1996.
- [32] Theo Odijk. Theory of lyotropic polymer liquid crystals. *Macromolecules*, 19(9):2313–2329, 1986.
- [33] R.B. Meyer, A. Ciferri, and WR Krigbaum. Polymer liquid crystals. *Academic, New York*, pages 133–164, 1982.
- [34] J. Hobdell and A. Windle. A numerical technique for predicting microstructure in liquid crystalline polymers. *Liquid crystals*, 23(2):157–173, 1997.

- [35] M. S Lavine. *Modelling of defect behaviour in nematic liquid crystalline materials*. PhD thesis, University of Cambridge, 1996.
- [36] M. Doi and S.F. Edwards. *The theory of polymer dynamics*, volume 73. Oxford University Press, USA, 1988.
- [37] R.G. Larson. On the relative magnitudes of viscous, elastic and texture stresses in liquid crystalline pbg solutions. *Rheologica acta*, 35(2):150–159, 1996.
- [38] R.G. Larson. Constitutive equations for polymer melts and solutions. *Butterworths*, 1988,, page 364, 1988.
- [39] PG De Gennes and J Prost. *The physics of liquid crystals*. 1993.
- [40] PG De Gennes and D Rainer. Alignment of anisotropic superfluids by flow. *Physics Letters A*, 46(7):429–430, 1974.
- [41] A. A. Sonin. *The surface physics of liquid crystals*. Taylor & Francis, 1995. ISBN 9782881249952.
- [42] AH Windle, HE Assender, MS Lavine, A. Donald, TCB McLeish, EL Thomas, AH Windle, HE Assender, MS Lavine, A. , et al. Modelling of form in thermotropic polymers [and discussion]. *Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences*, 348(1686):73–96, 1994.
- [43] AB Metzner and JC Reed. Flow of non-newtonian fluids correlation of the laminar, transition, and turbulent-flow regions. *AIChE Journal*, 1(4):434–440, 1955.

# Appendix A

## Introduction to UDFs

A user defined function (UDF) in ANSYS® FLUENT® is a piece of C code that can be compiled with the ANSYS® FLUENT® solver to add customized features and capabilities to the ANSYS® FLUENT® solver. Using the UDF is it possible to define custom boundary conditions, material properties, initial condition and so on.

Since these code are written in C programming language, it is possible to utilize all the powerful capabilities of the C programming and memory management. UDFs can be written in any text editor and then saved with `.c` extension. This C file can contain several different functions and UDFs. Using the UDFs, it is possible to reach the variables inside the solver data and manipulate this data for each iteration or time step.

In summary, UDFs:

- are written in C programming language (ANSYS® FLUENT® is also written in C Programming language.)
- must include `udf.h` in their header
- are defined using the **DEFINE** macro which is supplied by ANSYS® FLUENT®
- utilize predefined macros and functions to access the solver data and define functions to be assigned to different properties
- can be executed along with the solver when compiled
- are accessible in the graphical user interface of the ANSYS® FLUENT® to be assigned to properties
- like other C functions, return their value with return function (in SI unit)

## Appendix B

# Non-dimensional numbers

### 1. Reynolds number (Re)

- The ratio of the fluid inertia to viscous forces
- Newtonian fluid  $Re = (\rho V D)/\mu$
- Power-law fluid [43]

$$Re_{pl} = 2^{3-n} \left( \frac{n}{3n+1} \right)^n \frac{\bar{V}^{2-n} D^n \rho}{K}$$

### 2. Nusselt number (Nu)

- The ratio of convective to conductive heat transfer across (normal to) the boundary
- $Nu = h.L/k_f$
- Selection of the characteristic length,  $L$ , should be in the direction of growth (or thickness) of the boundary layer. Some examples of characteristic length are: the outer diameter of a cylinder in (external) cross flow (perpendicular to the cylinder axis), the length of a vertical plate undergoing natural convection, or the diameter of a sphere. For complex shapes, the length may be defined as the volume of the fluid body divided by the surface area. The thermal conductivity of the fluid is typically (but not always) evaluated at the film temperature, which for engineering purposes may be calculated as the mean-average of the bulk fluid temperature and wall surface temperature. For relations defined as a local Nusselt number, one should take the characteristic length to be the distance from the surface boundary to the local point of interest. However, to obtain an average

Nusselt number, one must integrate said relation over the entire characteristic length. Typically, for free convection, the average Nusselt number is expressed as a function of the Rayleigh number and the Prandtl number, written as:  $Nu = f(Ra, Pr)$ . Else, for forced convection, the Nusselt number is generally a function of the Reynolds number and the Prandtl number, or  $Nu = f(Re, Pr)$ . Empirical correlations for a wide variety of geometries are available that express the Nusselt number in the aforementioned forms.

### 3. Prandtl number (Pr)

- The ratio of momentum diffusivity (kinematic viscosity) to thermal diffusivity
- $Pr = \nu/\alpha = c_p\mu/k$

### 4. Weissenberg number (Wi)

- Shear rate times relaxation time
- $Wi = \dot{\gamma} \cdot \lambda$
- The Weissenberg number (Wi) is a dimensionless number used in the study of viscoelastic flows. It is named after Karl Weissenberg. The dimensionless number is the ratio of the relaxation time of the fluid and a specific process time. For instance, in simple steady shear, the Weissenberg number, often abbreviated as Wi or We, is defined as the shear rate times the relaxation time. Since this number is obtained from scaling the evolution of the stress, it contains choices for the shear or elongation rate, and the length-scale. Therefore the exact definition of all non dimensional numbers should be given as well as the number itself. While Wi is similar to the Deborah number and is often confused with it in technical literature, they have different physical interpretations. The Weissenberg number indicates the degree of anisotropy or orientation generated by the deformation, and is appropriate to describe flows with a constant stretch history, such as simple shear. In contrast, the Deborah number should be used to describe flows with a non-constant stretch history, and physically represents the rate at which elastic energy is stored or released.

### 5. Deborah number (De)

- The ratio of the stress relaxation time to the time scale of the observation
- $De = t_c/t_p$

- The Deborah number ( $De$ ) is a dimensionless number, often used in rheology to characterize the fluidity of materials under specific flow conditions. It is based on the premise that given enough time even the solid-like material will flow. The flow characteristics is not an inherent property of the material alone, but a relative property that depends on two fundamentally different characteristic times. Formally, the Deborah number is defined as the ratio of the relaxation time characterizing the time it takes for a material to adjust to applied stresses or deformations, and the characteristic time scale of an experiment (or a computer simulation) probing the response of the material. It incorporates both the elasticity and viscosity of the material. At lower Deborah numbers, the material behaves in a more fluidlike manner, with an associated Newtonian viscous flow. At higher Deborah numbers, the material behavior changes to a non-Newtonian regime, increasingly dominated by elasticity, demonstrating solidlike behavior.

## Appendix C

# Polymat's material data for LCP (MAZ078MMR2.0)

viscoelastic isothermal model

INITIAL DATA

nb. of modes = 1

mode # 1 - Phan Thien-Tanner model

$$T = T1 + T2$$

$$\exp(\text{eps} * \text{trelax} / \text{visc1} * \text{tr}(T1)) * T1 + \text{trelax} * ((1 - \text{xi}/2) * T1_{\text{up}} + \text{xi}/2 * T1_{\text{low}}) \\ = 2 * \text{visc1} * D$$

$$T2 = 2 * \text{visc2} * D$$

where - visc is the viscosity

- visc1 = (1-ratio)\*visc

- visc2 = ratio\*visc

- trelax is the relaxation time

- T1up is the upper-convected time derivative of T1

- T1low is the lower-convected time derivative of T1

visc = 0.4085412E+04 [auto]

trelax = 0.3162278E-02 [auto]  
eps = 0.8602864E-02 [auto]  
xi = 0.8534575E+00 [auto]  
ratio = 0.1711546E-02 [auto]

#### EXPERIMENTAL CURVES

nb. of experimental curves : 3

curve #0 : name = Inputs\Shear.crv  
- temperature .... : 3.000000e+002  
- nb. of points .. : 16

shear rate,	steady shear viscosity
5.5000000e+000	1.1383000e+003
5.5000000e+000	1.1190000e+003
1.1270000e+001	5.6470000e+002
1.2090000e+001	5.3490000e+002
1.0970000e+002	1.9820000e+002
1.0970000e+002	2.0790000e+002
4.9291000e+002	1.0260000e+002
4.9566000e+002	1.0060000e+002
9.8942000e+002	7.1100000e+001
9.9052000e+002	7.0800000e+001
5.0028800e+003	2.8500000e+001
5.0031500e+003	2.8400000e+001
7.5037900e+003	2.1900000e+001
7.5046200e+003	2.2300000e+001
1.0004680e+004	1.8500000e+001
1.0005510e+004	1.8400000e+001

curve #1 : name = Inputs\GPrime.crv  
- temperature .... : 3.000000e+002  
- nb. of points .. : 31



frequence,	storage modulus G'
1.0000000e-001	3.7305000e+002
1.2590000e-001	3.9810000e+002
1.5850000e-001	4.9790000e+002
1.9950000e-001	5.1435000e+002
2.5120000e-001	5.6640000e+002
3.1620000e-001	6.0675000e+002
3.9810000e-001	6.7675000e+002
5.0120000e-001	7.3665000e+002
6.3100000e-001	7.7960000e+002
7.9430000e-001	8.5195000e+002
1.0000000e+000	8.9855000e+002
1.2590000e+000	9.4360000e+002
1.5850000e+000	1.0415500e+003
1.9950000e+000	1.1380000e+003
2.5120000e+000	1.2425000e+003
3.1620000e+000	1.3445000e+003
3.9810000e+000	1.4700000e+003
5.0120000e+000	1.6300000e+003
6.3100000e+000	1.7815000e+003
7.9430000e+000	1.9520000e+003
1.0000000e+001	2.1730000e+003
1.2590000e+001	2.4290000e+003
1.5850000e+001	2.6960000e+003
1.9950000e+001	3.0180000e+003
2.5120000e+001	3.3840000e+003
3.1620000e+001	3.8340000e+003
3.9810000e+001	4.3515000e+003
5.0120000e+001	4.9425000e+003
6.3100000e+001	5.6150000e+003
7.9430000e+001	6.3865000e+003
1.0000000e+002	7.3205000e+003

curve #2 : name = Inputs\GDoublePrime.crv  
- temperature .... : 3.000000e+002

- nb. of points .. : 31

frequence,	loss modulus G''
1.000000e-001	2.156000e+002
1.259000e-001	2.356500e+002
1.585000e-001	2.793000e+002
1.995000e-001	3.489500e+002
2.512000e-001	3.991000e+002
3.162000e-001	4.159000e+002
3.981000e-001	4.625000e+002
5.012000e-001	5.269500e+002
6.310000e-001	5.263500e+002
7.943000e-001	5.597500e+002
1.000000e+000	6.621000e+002
1.259000e+000	7.297500e+002
1.585000e+000	8.116000e+002
1.995000e+000	9.337000e+002
2.512000e+000	1.034900e+003
3.162000e+000	1.167500e+003
3.981000e+000	1.335000e+003
5.012000e+000	1.524500e+003
6.310000e+000	1.741000e+003
7.943000e+000	1.993500e+003
1.000000e+001	2.274500e+003
1.259000e+001	2.618000e+003
1.585000e+001	2.995000e+003
1.995000e+001	3.443000e+003
2.512000e+001	3.936000e+003
3.162000e+001	4.501000e+003
3.981000e+001	5.143000e+003
5.012000e+001	5.900000e+003
6.310000e+001	6.749000e+003
7.943000e+001	7.665000e+003
1.000000e+002	8.757500e+003

## FITTING

max. nb. of iterations = 50

Step 0 : Evaluation of the relaxation times

mode 1 : relaxation time = 3.1622777e-003

Step 1 : Evaluation of the partial viscosities

Distance between two successive solutions,	Distance between solution and experimental points
---	--

1.8987342e+000	2.2836354e+001
----------------	----------------

0.0000000e+000	2.2836354e+001
----------------	----------------

Number of iterations : 2

-> Optimisation terminated

mode 1 : partial viscosity = 4.3944597e+003

Step 2 : Evaluation of other parameters

Distance between two successive solutions,	Distance between solution and experimental points
---	--

1.8183540e+000	4.4476405e+001
----------------	----------------

1.6536446e+000	4.3733101e+001
----------------	----------------

6.8400870e-001	4.3664110e+001
----------------	----------------

Number of iterations : 3

-> Optimisation terminated

## RESULTS

nb. of modes = 1

mode # 1 - Phan Thien-Tanner model

$$T = T1 + T2$$

$$\begin{aligned} \exp(\text{eps} * \text{trelax} / \text{visc1} * \text{tr}(T1)) * T1 + \text{trelax} * ((1 - \text{xi} / 2) * T1_{\text{up}} + \text{xi} / 2 * T1_{\text{low}}) \\ = 2 * \text{visc1} * D \end{aligned}$$

$$T2 = 2 * \text{visc2} * D$$

where - visc is the viscosity

$$- \text{visc1} = (1 - \text{ratio}) * \text{visc}$$

$$- \text{visc2} = \text{ratio} * \text{visc}$$

- trelax is the relaxation time

- T1up is the upper-convected time derivative of T1

- T1low is the lower-convected time derivative of T1

visc = 0.4085412E+04 [auto]

trelax = 0.3162278E-02 [auto]

eps = 0.8602864E-02 [auto]

xi = 0.8534575E+00 [auto]

ratio = 0.1711546E-02 [auto]

ELAPSED time : 0 s

CPU time ... : 0 s

## Appendix D

# UDF code for simulating the directionality

```
#include "stdio.h"
#include "stdlib.h"
#include "udf.h"
#include "math.h"
#include "mem.h"
#include "sg.h"

DEFINE_ON_DEMAND(orientation)
{
    Domain *domain=Get_Domain(1);
    Thread *t;
    cell_t c;
    int i;
    thread_loop_c(t,domain)/*This part sets all the UDMs to zero*/
    {
        begin_c_loop(c,t)
        {
            #if RP_2D
                C_UDMI(c,t,0)=0.0;
                C_UDMI(c,t,1)=0.0;
                C_UDMI(c,t,2)=0.0;
            #endif
        }
    }
}
```

```

    C_UDMI(c,t,3)=0.0;
    C_UDMI(c,t,4)=0.0;
    C_UDMI(c,t,5)=0.0;

#endif
#if RP_3D
    C_UDMI(c,t,0)=0.0;
    C_UDMI(c,t,1)=0.0;
    C_UDMI(c,t,2)=0.0;
    C_UDMI(c,t,3)=0.0;
    C_UDMI(c,t,4)=0.0;
    C_UDMI(c,t,5)=0.0;
    C_UDMI(c,t,6)=0.0;
    C_UDMI(c,t,7)=0.0;
    C_UDMI(c,t,8)=0.0;
#endif
}
end_c_loop(c,t)
}

Random(); /*This function generates a random distribution of
    directors*/

//Initial_condition();

for (i=0;i<20;i++) /*Calculating the evolution order parameter*/
{
// Message(" Iteration: %i is done.\n", i);

    thread_loop_c(t,domain)/*This part sets all the UDMs for
        calculation to zero*/
    {
        begin_c_loop(c,t)
        {
#if RP_2D

```

```

    C_UDMI(c,t,3)=0.0;
    C_UDMI(c,t,4)=0.0;
    C_UDMI(c,t,5)=0.0;

#endif
#if RP_3D
    C_UDMI(c,t,5)=0.0;
    C_UDMI(c,t,6)=0.0;
    C_UDMI(c,t,7)=0.0;
    C_UDMI(c,t,8)=0.0;
#endif
}
end_c_loop(c,t)
}

Evolution();/*This function applys the effect of evolution
equation to the directors
and saves the result in the following UDMs*/
// Message("Evolution Equation: %i is done.\n", i);
thread_loop_c(t,domain) /*This part assigns the calculated
directors in the Evolution()
subroutine to the user defined memories for the n_x, n_y and in
case of 3D n_z*/
{
begin_c_loop(c,t)
{
#if RP_2D
C_UDMI(c,t,1)+=C_UDMI(c,t,3);
C_UDMI(c,t,2)+=C_UDMI(c,t,4);
if ((C_UDMI(c,t,1)>0) && (C_UDMI(c,t,2)<0))
{
C_UDMI(c,t,1)*=-1.0;
C_UDMI(c,t,2)*=-1.0;
}
if ((C_UDMI(c,t,1)<0) && (C_UDMI(c,t,2)<0))

```

```

{
C_UDMI(c,t,1)*=-1.0;
C_UDMI(c,t,2)*=-1.0;
}

C_UDMI(c,t,1)=C_UDMI(c,t,1)/sqrt(pow(C_UDMI(c,t,1),2)+pow(
    C_UDMI(c,t,2),2));
C_UDMI(c,t,2)=C_UDMI(c,t,2)/sqrt(pow(C_UDMI(c,t,1),2)+pow(
    C_UDMI(c,t,2),2));

#endif
#if RP_3D

C_UDMI(c,t,2)+=C_UDMI(c,t,5);
C_UDMI(c,t,3)+=C_UDMI(c,t,6);
C_UDMI(c,t,4)+=C_UDMI(c,t,7);
/*This part applies the fact that directors have sence but not
    direction*/
if ((C_UDMI(c,t,2)>0) && (C_UDMI(c,t,3)<0) && (C_UDMI(c,t,4)>0)
    )
{
C_UDMI(c,t,2)*=-1.0;
C_UDMI(c,t,3)*=-1.0;
C_UDMI(c,t,4)*=-1.0;
}
if ((C_UDMI(c,t,2)>0) && (C_UDMI(c,t,3)<0) && (C_UDMI(c,t,4)<0)
    )
{
C_UDMI(c,t,2)*=-1.0;
C_UDMI(c,t,3)*=-1.0;
C_UDMI(c,t,4)*=-1.0;
}
if ((C_UDMI(c,t,2)<0) && (C_UDMI(c,t,3)<0) && (C_UDMI(c,t,4)>0)
    )
{

```



```

C_UDMI(c,t,2)*=-1.0;
C_UDMI(c,t,3)*=-1.0;
C_UDMI(c,t,4)*=-1.0;
}
if ((C_UDMI(c,t,2)<0) && (C_UDMI(c,t,3)<0) && (C_UDMI(c,t,4)<0)
    )
{
C_UDMI(c,t,2)*=-1.0;
C_UDMI(c,t,3)*=-1.0;
C_UDMI(c,t,4)*=-1.0;
}

C_UDMI(c,t,2)=C_UDMI(c,t,2)/sqrt(pow(C_UDMI(c,t,2),2)+pow(
    C_UDMI(c,t,3),2)+pow(C_UDMI(c,t,4),2));
C_UDMI(c,t,3)=C_UDMI(c,t,3)/sqrt(pow(C_UDMI(c,t,2),2)+pow(
    C_UDMI(c,t,3),2)+pow(C_UDMI(c,t,4),2));
C_UDMI(c,t,4)=C_UDMI(c,t,4)/sqrt(pow(C_UDMI(c,t,2),2)+pow(
    C_UDMI(c,t,3),2)+pow(C_UDMI(c,t,4),2));

#endif
}
end_c_loop(c,t)
}
}
OrderParameter();

thread_loop_c(t,domain)/*This part sets all the UDMs to zero*/
{
begin_c_loop(c,t)
{
#if RP_2D
C_UDMI(c,t,0)=0.0;
C_UDMI(c,t,1)=0.0;
C_UDMI(c,t,2)=0.0;
C_UDMI(c,t,3)=0.0;

```

```

    C_UDMI(c,t,4)=0.0;
    C_UDMI(c,t,5)=0.0;

#endif
#if RP_3D
    C_UDMI(c,t,0)=0.0;
    C_UDMI(c,t,1)=0.0;
    C_UDMI(c,t,2)=0.0;
    C_UDMI(c,t,3)=0.0;
    C_UDMI(c,t,4)=0.0;
    C_UDMI(c,t,5)=0.0;
    C_UDMI(c,t,6)=0.0;
    C_UDMI(c,t,7)=0.0;
    C_UDMI(c,t,8)=0.0;
#endif
}
end_c_loop(c,t)
}

Random(); /*This function generates a random distribution of
    directors*/

for (i=0;i<20;i++) /*This part calculates the Franks order
    parameter*/
{
// Message(" Iteration: %i is done.\n", i);

thread_loop_c(t,domain)/*This part sets all the UDMs for
    calculation to zero*/
{
begin_c_loop(c,t)
{
#if RP_2D
    C_UDMI(c,t,3)=0.0;
    C_UDMI(c,t,4)=0.0;

```

```

        C_UDMI(c,t,5)=0.0;

#endif
#if RP_3D
    C_UDMI(c,t,5)=0.0;
    C_UDMI(c,t,6)=0.0;
    C_UDMI(c,t,7)=0.0;
    C_UDMI(c,t,8)=0.0;
#endif
    }
    end_c_loop(c,t)
}

Frank(); /* This function calculates the effect of Franks
        elastic equation on the
        directors and saves the results in the following UDMs. */

// Message("Franks Equation: %i is done.\n", i);

thread_loop_c(t, domain)
{
begin_c_loop(c,t)
{
#if RP_2D
C_UDMI(c,t,1)+=C_UDMI(c,t,3);
C_UDMI(c,t,2)+=C_UDMI(c,t,4);

if ((C_UDMI(c,t,1)>0) && (C_UDMI(c,t,2)<0))
{
C_UDMI(c,t,1)*=-1.0;
C_UDMI(c,t,2)*=-1.0;
}
if ((C_UDMI(c,t,1)<0) && (C_UDMI(c,t,2)<0))
{

```

```

C_UDMI(c,t,1)*=-1.0;
C_UDMI(c,t,2)*=-1.0;
}

C_UDMI(c,t,1)=C_UDMI(c,t,1)/sqrt(pow(C_UDMI(c,t,1),2)+pow(
    C_UDMI(c,t,2),2));
C_UDMI(c,t,2)=C_UDMI(c,t,2)/sqrt(pow(C_UDMI(c,t,1),2)+pow(
    C_UDMI(c,t,2),2));
#endif

#if RP_3D

C_UDMI(c,t,2)+=C_UDMI(c,t,5);
C_UDMI(c,t,3)+=C_UDMI(c,t,6);
C_UDMI(c,t,4)+=C_UDMI(c,t,7);

if ((C_UDMI(c,t,2)>0) && (C_UDMI(c,t,3)<0) && (C_UDMI(c,t,4)>0)
    )
{
C_UDMI(c,t,2)*=-1.0;
C_UDMI(c,t,3)*=-1.0;
C_UDMI(c,t,4)*=-1.0;
}
if ((C_UDMI(c,t,2)>0) && (C_UDMI(c,t,3)<0) && (C_UDMI(c,t,4)<0)
    )
{
C_UDMI(c,t,2)*=-1.0;
C_UDMI(c,t,3)*=-1.0;
C_UDMI(c,t,4)*=-1.0;
}
if ((C_UDMI(c,t,2)<0) && (C_UDMI(c,t,3)<0) && (C_UDMI(c,t,4)>0)
    )
{
C_UDMI(c,t,2)*=-1.0;
C_UDMI(c,t,3)*=-1.0;

```

```

C_UDMI(c,t,4)*=-1.0;
}
if ((C_UDMI(c,t,2)<0) && (C_UDMI(c,t,3)<0) && (C_UDMI(c,t,4)<0)
    )
{
C_UDMI(c,t,2)*=-1.0;
C_UDMI(c,t,3)*=-1.0;
C_UDMI(c,t,4)*=-1.0;
}

C_UDMI(c,t,2)=C_UDMI(c,t,2)/sqrt(pow(C_UDMI(c,t,2),2)+pow(
    C_UDMI(c,t,3),2)+pow(C_UDMI(c,t,4),2));
C_UDMI(c,t,3)=C_UDMI(c,t,3)/sqrt(pow(C_UDMI(c,t,2),2)+pow(
    C_UDMI(c,t,3),2)+pow(C_UDMI(c,t,4),2));
C_UDMI(c,t,4)=C_UDMI(c,t,4)/sqrt(pow(C_UDMI(c,t,2),2)+pow(
    C_UDMI(c,t,3),2)+pow(C_UDMI(c,t,4),2));

#endif
}
end_c_loop(c,t)
}

OrderParameter();

thread_loop_c(t,domain)/*This part sets all the UDMs to zero*/
{
begin_c_loop(c,t)
{
#if RP_2D
C_UDMI(c,t,0)=0.0;
C_UDMI(c,t,1)=0.0;
C_UDMI(c,t,2)=0.0;
C_UDMI(c,t,3)=0.0;
C_UDMI(c,t,4)=0.0;

```

```

    C_UDMI(c,t,5)=0.0;

#endif
#if RP_3D
    C_UDMI(c,t,0)=0.0;
    C_UDMI(c,t,1)=0.0;
    C_UDMI(c,t,2)=0.0;
    C_UDMI(c,t,3)=0.0;
    C_UDMI(c,t,4)=0.0;
    C_UDMI(c,t,5)=0.0;
    C_UDMI(c,t,6)=0.0;
    C_UDMI(c,t,7)=0.0;
    C_UDMI(c,t,8)=0.0;
#endif
}
end_c_loop(c,t)
}

Random(); /*This function generates a random distribution of
    directors*/

for (i=0;i<20;i++) /*This part calculates the order parameter
    for the combination of all three effects*/
{
// Message(" Itteration: %i is done.\n", i);

    thread_loop_c(t,domain)/*This part sets all the UDMs for
        calculation to zero*/
    {
        begin_c_loop(c,t)
        {
#if RP_2D
            C_UDMI(c,t,3)=0.0;
            C_UDMI(c,t,4)=0.0;
            C_UDMI(c,t,5)=0.0;

```

```

#endif
#if RP_3D
    C_UDMI(c,t,5)=0.0;
    C_UDMI(c,t,6)=0.0;
    C_UDMI(c,t,7)=0.0;
    C_UDMI(c,t,8)=0.0;
#endif
}
end_c_loop(c,t)
}

Evolution();/*This function applies the effect of evolution
equation to the directors
and saves the result in the following UDMs*/
// Message("Evolution Equation: %i is done.\n", i);
thread_loop_c(t,domain) /*This part assigns the calculated
directors in the Evolution()
subroutine to the user defined memories for the n_x, n_y and in
case of 3D n_z*/
{
begin_c_loop(c,t)
{
#if RP_2D
C_UDMI(c,t,1)+=C_UDMI(c,t,3);
C_UDMI(c,t,2)+=C_UDMI(c,t,4);
if ((C_UDMI(c,t,1)>0) && (C_UDMI(c,t,2)<0))
{
C_UDMI(c,t,1)*=-1.0;
C_UDMI(c,t,2)*=-1.0;
}
if ((C_UDMI(c,t,1)<0) && (C_UDMI(c,t,2)<0))
{
C_UDMI(c,t,1)*=-1.0;
C_UDMI(c,t,2)*=-1.0;
}
}
}

```

```

}

C_UDMI(c,t,1)=C_UDMI(c,t,1)/sqrt(pow(C_UDMI(c,t,1),2)+pow(
    C_UDMI(c,t,2),2));
C_UDMI(c,t,2)=C_UDMI(c,t,2)/sqrt(pow(C_UDMI(c,t,1),2)+pow(
    C_UDMI(c,t,2),2));

#endif
#if RP_3D

C_UDMI(c,t,2)+=C_UDMI(c,t,5);
C_UDMI(c,t,3)+=C_UDMI(c,t,6);
C_UDMI(c,t,4)+=C_UDMI(c,t,7);
/*This part applys the fact that directors have sence but not
    direction*/
if ((C_UDMI(c,t,2)>0) && (C_UDMI(c,t,3)<0) && (C_UDMI(c,t,4)>0)
    )
{
C_UDMI(c,t,2)*=-1.0;
C_UDMI(c,t,3)*=-1.0;
C_UDMI(c,t,4)*=-1.0;
}
if ((C_UDMI(c,t,2)>0) && (C_UDMI(c,t,3)<0) && (C_UDMI(c,t,4)<0)
    )
{
C_UDMI(c,t,2)*=-1.0;
C_UDMI(c,t,3)*=-1.0;
C_UDMI(c,t,4)*=-1.0;
}
if ((C_UDMI(c,t,2)<0) && (C_UDMI(c,t,3)<0) && (C_UDMI(c,t,4)>0)
    )
{
C_UDMI(c,t,2)*=-1.0;
C_UDMI(c,t,3)*=-1.0;
C_UDMI(c,t,4)*=-1.0;
}

```



```

}
if ((C_UDMI(c,t,2)<0) && (C_UDMI(c,t,3)<0) && (C_UDMI(c,t,4)<0)
    )
{
C_UDMI(c,t,2)*=-1.0;
C_UDMI(c,t,3)*=-1.0;
C_UDMI(c,t,4)*=-1.0;
}

C_UDMI(c,t,2)=C_UDMI(c,t,2)/sqrt(pow(C_UDMI(c,t,2),2)+pow(
    C_UDMI(c,t,3),2)+pow(C_UDMI(c,t,4),2));
C_UDMI(c,t,3)=C_UDMI(c,t,3)/sqrt(pow(C_UDMI(c,t,2),2)+pow(
    C_UDMI(c,t,3),2)+pow(C_UDMI(c,t,4),2));
C_UDMI(c,t,4)=C_UDMI(c,t,4)/sqrt(pow(C_UDMI(c,t,2),2)+pow(
    C_UDMI(c,t,3),2)+pow(C_UDMI(c,t,4),2));

#endif
}
end_c_loop(c,t)
}

thread_loop_c(t,domain)/*This part sets all the UDMs for
    calculation to zero*/
{
begin_c_loop(c,t)
{
#if RP_2D
C_UDMI(c,t,3)=0.0;
C_UDMI(c,t,4)=0.0;
C_UDMI(c,t,5)=0.0;

#endif

#if RP_3D
C_UDMI(c,t,5)=0.0;
C_UDMI(c,t,6)=0.0;

```

```

C_UDMI(c,t,7)=0.0;
C_UDMI(c,t,8)=0.0;
#endif
}
end_c_loop(c,t)
}

Frank(); /* This function calculates the effect of Franks
         elastic equation on the
         directors and saves the results in the following UDMs. */

// Message("Franks Equation: %i is done.\n", i);

thread_loop_c(t, domain)
{
begin_c_loop(c,t)
{
#if RP_2D
C_UDMI(c,t,1)+=C_UDMI(c,t,3);
C_UDMI(c,t,2)+=C_UDMI(c,t,4);

if ((C_UDMI(c,t,1)>0) && (C_UDMI(c,t,2)<0))
{
C_UDMI(c,t,1)*=-1.0;
C_UDMI(c,t,2)*=-1.0;
}
if ((C_UDMI(c,t,1)<0) && (C_UDMI(c,t,2)<0))
{
C_UDMI(c,t,1)*=-1.0;
C_UDMI(c,t,2)*=-1.0;
}

C_UDMI(c,t,1)=C_UDMI(c,t,1)/sqrt(pow(C_UDMI(c,t,1),2)+pow(
C_UDMI(c,t,2),2));

```

```

C_UDMI(c,t,2)=C_UDMI(c,t,2)/sqrt(pow(C_UDMI(c,t,1),2)+pow(
    C_UDMI(c,t,2),2));
#endif

#if RP_3D

C_UDMI(c,t,2)+=C_UDMI(c,t,5);
C_UDMI(c,t,3)+=C_UDMI(c,t,6);
C_UDMI(c,t,4)+=C_UDMI(c,t,7);

if ((C_UDMI(c,t,2)>0) && (C_UDMI(c,t,3)<0) && (C_UDMI(c,t,4)>0)
    )
{
C_UDMI(c,t,2)*=-1.0;
C_UDMI(c,t,3)*=-1.0;
C_UDMI(c,t,4)*=-1.0;
}
if ((C_UDMI(c,t,2)>0) && (C_UDMI(c,t,3)<0) && (C_UDMI(c,t,4)<0)
    )
{
C_UDMI(c,t,2)*=-1.0;
C_UDMI(c,t,3)*=-1.0;
C_UDMI(c,t,4)*=-1.0;
}
if ((C_UDMI(c,t,2)<0) && (C_UDMI(c,t,3)<0) && (C_UDMI(c,t,4)>0)
    )
{
C_UDMI(c,t,2)*=-1.0;
C_UDMI(c,t,3)*=-1.0;
C_UDMI(c,t,4)*=-1.0;
}
if ((C_UDMI(c,t,2)<0) && (C_UDMI(c,t,3)<0) && (C_UDMI(c,t,4)<0)
    )
{
C_UDMI(c,t,2)*=-1.0;

```

```

C_UDMI(c,t,3)*=-1.0;
C_UDMI(c,t,4)*=-1.0;
}

```

```

C_UDMI(c,t,2)=C_UDMI(c,t,2)/sqrt(pow(C_UDMI(c,t,2),2)+pow(
    C_UDMI(c,t,3),2)+pow(C_UDMI(c,t,4),2));
C_UDMI(c,t,3)=C_UDMI(c,t,3)/sqrt(pow(C_UDMI(c,t,2),2)+pow(
    C_UDMI(c,t,3),2)+pow(C_UDMI(c,t,4),2));
C_UDMI(c,t,4)=C_UDMI(c,t,4)/sqrt(pow(C_UDMI(c,t,2),2)+pow(
    C_UDMI(c,t,3),2)+pow(C_UDMI(c,t,4),2));

```

```

#endif
}
end_c_loop(c,t)
}

```

```

Translation();
thread_loop_c(t,domain)
{
    begin_c_loop(c,t)
    {

```

```

#if RP_2D
    if(C_UDMI(c,t,5)>0)
    {
        //    C_UDMI(c,t,1)+=C_UDMI(c,t,3);
        //    C_UDMI(c,t,2)+=C_UDMI(c,t,4);
        C_UDMI(c,t,1)=C_UDMI(c,t,3)/sqrt(pow(C_UDMI(c,t,3),2)+pow(
            C_UDMI(c,t,4),2));
        C_UDMI(c,t,2)=C_UDMI(c,t,4)/sqrt(pow(C_UDMI(c,t,3),2)+pow(
            C_UDMI(c,t,4),2));
        if ((C_UDMI(c,t,1)>0) && (C_UDMI(c,t,2)<0))
        {
            C_UDMI(c,t,1)*=-1.0;
            C_UDMI(c,t,2)*=-1.0;

```

```

}
if ((C_UDMI(c,t,1)<0) && (C_UDMI(c,t,2)<0))
{
    C_UDMI(c,t,1)*=-1.0;
    C_UDMI(c,t,2)*=-1.0;
}
}
else if (C_UDMI(c,t,5)==0)
{ /*This happens on the boundaries of the flow domain*/

    C_UDMI(c,t,0)=(rand()%31416)/10000.0;
    C_UDMI(c,t,1)=cos(C_UDMI(c,t,0)); /*x component of the
        director*/
    C_UDMI(c,t,2)=sin(C_UDMI(c,t,0)); /*y component of the
        director*/
}
#endif

#if RP_3D
if (C_UDMI(c,t,8)>0)
{
    //    C_UDMI(c,t,2)+=C_UDMI(c,t,5);
    //    C_UDMI(c,t,3)+=C_UDMI(c,t,6);
    //    C_UDMI(c,t,4)+=C_UDMI(c,t,7);
    C_UDMI(c,t,2)=C_UDMI(c,t,5)/sqrt(pow(C_UDMI(c,t,5),2)+pow(
        C_UDMI(c,t,6),2)+pow(C_UDMI(c,t,7),2));
    C_UDMI(c,t,3)=C_UDMI(c,t,6)/sqrt(pow(C_UDMI(c,t,5),2)+pow(
        C_UDMI(c,t,6),2)+pow(C_UDMI(c,t,7),2));
    C_UDMI(c,t,4)=C_UDMI(c,t,7)/sqrt(pow(C_UDMI(c,t,5),2)+pow(
        C_UDMI(c,t,6),2)+pow(C_UDMI(c,t,7),2));
    if ((C_UDMI(c,t,2)>0) && (C_UDMI(c,t,3)<0) && (C_UDMI(c,t,4)
        >0))
    {
        C_UDMI(c,t,2)*=-1.0;
        C_UDMI(c,t,3)*=-1.0;
    }
}
}

```

```

    C_UDMI(c , t , 4) *=-1.0;
}
if ((C_UDMI(c , t , 2)>0) && (C_UDMI(c , t , 3)<0) && (C_UDMI(c , t , 4)
    <0))
{
    C_UDMI(c , t , 2) *=-1.0;
    C_UDMI(c , t , 3) *=-1.0;
    C_UDMI(c , t , 4) *=-1.0;
}
if ((C_UDMI(c , t , 2)<0) && (C_UDMI(c , t , 3)<0) && (C_UDMI(c , t , 4)
    >0))
{
    C_UDMI(c , t , 2) *=-1.0;
    C_UDMI(c , t , 3) *=-1.0;
    C_UDMI(c , t , 4) *=-1.0;
}
if ((C_UDMI(c , t , 2)<0) && (C_UDMI(c , t , 3)<0) && (C_UDMI(c , t , 4)
    <0))
{
    C_UDMI(c , t , 2) *=-1.0;
    C_UDMI(c , t , 3) *=-1.0;
    C_UDMI(c , t , 4) *=-1.0;
}
}
else if (C_UDMI(c , t , 8)==0)
{
    C_UDMI(c , t , 0)=(rand () %31416) /10000.0;
    C_UDMI(c , t , 1)=(rand () %31416) /10000.0;
    C_UDMI(c , t , 2)=sin (C_UDMI(c , t , 1) ) *cos (C_UDMI(c , t , 0) ) ; /*x
        component of the director*/
    C_UDMI(c , t , 3)=sin (C_UDMI(c , t , 1) ) *sin (C_UDMI(c , t , 0) ) ; /*y
        component of the director*/
    C_UDMI(c , t , 4)=cos (C_UDMI(c , t , 1) ) ; /*z component of the
        director*/
}

```

```

#endif

    }
    end_c_loop(c,t)
}
}
OrderParameter();

Message(" Done.\n");
}

void Random()
{
Domain *domain=Get_Domain(1);
Thread *t;
cell_t c;
thread_loop_c(t, domain)
{
begin_c_loop(c,t)
{
C_UDMI(c,t,0)=(rand()%31416)/10000.0;

#if RP_2D
Set_User_Memory_Name(0," angle_from_x_axis");
Set_User_Memory_Name(1," directors_x_component");
Set_User_Memory_Name(2," directors_y_component");
C_UDMI(c,t,1)=cos(C_UDMI(c,t,0));/*x component of the director
*/
C_UDMI(c,t,2)=sin(C_UDMI(c,t,0));/*y component of the director
*/
#endif

#if RP_3D
C_UDMI(c,t,1)=(rand()%31416)/10000.0;

```

```

Set_User_Memory_Name(0," angle_from_x_axis");
Set_User_Memory_Name(1," angle_from_y_axis");
Set_User_Memory_Name(2," directors_x_component");
Set_User_Memory_Name(3," directors_y_component");
Set_User_Memory_Name(4," directors_z_component");
C_UDMI(c,t,2)=sin(C_UDMI(c,t,1))*cos(C_UDMI(c,t,0));/*x
    component of the director*/
C_UDMI(c,t,3)=sin(C_UDMI(c,t,1))*sin(C_UDMI(c,t,0));/*y
    component of the director*/
C_UDMI(c,t,4)=cos(C_UDMI(c,t,1));/*z component of the director
    */
#endif

}
end_c_loop(c,t)
}
}

void Evolution()
{
/* the function (Evolution) will apply the effect of rate of
    rotation and rate
of deformation tensors to the director of each cell in the
    domain. This function
used the deterministic approach to apply the change in director
    Since this function uses the derivatives of the functions from
    FLUENT, these
derivatives should be stored and available to the function at
    the time of the
simulation. */
Domain *domain=Get_Domain(1);
Thread *t;
cell_t c;
face_t f;

```



```

double Deltat=1.e-4,Lambda=1.05 ,nx ,ny ,nz ,AA1,AA2,AA3,max=-10.0,
    min=-10.0;
double A[ND_ND][ND_ND],W[ND_ND][ND_ND],DeltaNW[ND_ND],DeltaNA[
    ND_ND];
int i;
thread_loop_c(t, domain)
{
    begin_c_loop(c, t)
    {
        A[0][0]=(-1.0)*CDUDX(c, t);
        A[0][1]=0.5*(CDUDY(c, t)+CDVDX(c, t));
        A[1][0]=A[0][1];
        A[1][1]=(-1.0)*CDVDY(c, t);
        W[0][0]=0.0;
        W[0][1]=0.5*(CDUDY(c, t)-CDVDX(c, t));
        W[1][0]=(-1.0)*W[0][1]; /*0.5*(CDVDX(c, t)-CDUDY(c, t));*/
        W[1][1]=0.0;
#ifdef RP_2D
        nx=C_UDMI(c, t, 1);
        ny=C_UDMI(c, t, 2);
#endif

#ifdef RP_3D
        A[0][2]=0.5*(CDUDZ(c, t)+CDWDX(c, t));
        A[1][2]=0.5*(CDVDZ(c, t)+CDWDY(c, t));
        A[2][2]=CDWDZ(c, t);
        A[2][1]=A[1][2];
        A[2][0]=A[0][2];
        W[0][2]=0.5*(CDUDZ(c, t)-CDWDX(c, t));
        W[1][2]=0.5*(CDVDZ(c, t)-CDWDY(c, t));
        W[2][2]=0.0;
        W[2][1]=(-1.0)*W[1][2]; /*0.5*(CDWDY(c, t)-CDVDZ(c, t));*/
        W[2][0]=(-1.0)*W[0][2]; /*0.5*(CDWDX(c, t)-CDUDZ(c, t));*/
        nx=C_UDMI(c, t, 2);
        ny=C_UDMI(c, t, 3);
#endif
    }
}

```

```

    nz=C_UDMI(c,t,4);
#endif

    DeltaNW[0]=Deltat*(nx*W[0][0]+ny*W[0][1]);
    DeltaNW[1]=Deltat*(nx*W[1][0]+ny*W[1][1]);

#if RP_3D
    DeltaNW[0]+=Deltat*(nz*W[0][2]);
    DeltaNW[1]+=Deltat*(nz*W[1][2]);
    DeltaNW[2]=Deltat*(nx*W[2][0]+ny*W[2][1]+nz*W[2][2]);
#endif

#if RP_2D
    DeltaNA[0]=(nx*A[0][0]+ny*A[1][0]-(nx*(nx*A[0][0]+ny*A[1][0])+
        ny*(nx*A[0][1]+ny*A[1][1]))*nx)*Deltat;
    DeltaNA[1]=(nx*A[0][1]+ny*A[1][1]-(nx*(nx*A[0][0]+ny*A[1][0])+
        ny*(nx*A[0][1]+ny*A[1][1]))*ny)*Deltat;
#endif

#if RP_3D
    DeltaNA[0]=(nx*A[0][0]+ny*A[1][0]+nz*A[2][0]-(nx*(nx*A[0][0]+
        ny*A[1][0]+nz*A[2][0])+ny*(nx*A[0][1]+ny*A[1][1]+nz*A
        [2][1])+nz*(nx*A[0][2]+ny*A[1][2]+nz*A[2][2])))*Deltat;
    DeltaNA[1]=(nx*A[0][1]+ny*A[1][1]+nz*A[2][1]-(nx*(nx*A[0][0]+
        ny*A[1][0]+nz*A[2][0])+ny*(nx*A[0][1]+ny*A[1][1]+nz*A
        [2][1])+nz*(nx*A[0][2]+ny*A[1][2]+nz*A[2][2])))*Deltat;
    DeltaNA[2]=(nx*A[0][2]+ny*A[1][2]+nz*A[2][2]-(nx*(nx*A[0][0]+
        ny*A[1][0]+nz*A[2][0])+ny*(nx*A[0][1]+ny*A[1][1]+nz*A
        [2][1])+nz*(nx*A[0][2]+ny*A[1][2]+nz*A[2][2])))*Deltat;
#endif

#if RP_2D
    /*The effects of evolution equation are stored in UDMs after
       the director components*/
    C_UDMI(c,t,3)=DeltaNW[0]+Lambda*DeltaNA[0];

```

```

C_UDMI(c,t,4)=DeltaNW[1]+Lambda*DeltaNA[1];

if ((C_UDMI(c,t,3) > 2.0))
{
  Message("C_UDMI(c,t,3) is: %e and C_UDMI(c,t,4) is: %e .\n",
    C_UDMI(c,t,3) ,C_UDMI(c,t,4) );
}

if (C_UDMI(c,t,3) == 0.0)
{
  Message("C_UDMI(c,t,3)= %e and C_UDMI(c,t,4)= %e .\n", C_UDMI
    (c,t,3) ,C_UDMI(c,t,4) );
}
/*
if (sqrt(pow(C_UDMI(c,t,3),2)+pow(C_UDMI(c,t,4),2))>0.1)
{
  Message("Deltat is too large and the magnitude of dn is larger
    than 0.1.\n");
  Message("New director 's length for cell %i is: %f \n",c,sqrt(
    pow(C_UDMI(c,t,3),2)+pow(C_UDMI(c,t,4),2)));
}*/

#endif

#if RP_3D
  /*The effects of evolution equation are stored in UDMs after
    the director components*/
  C_UDMI(c,t,5)=DeltaNW[0]+Lambda*DeltaNA[0];
  C_UDMI(c,t,6)=DeltaNW[1]+Lambda*DeltaNA[1];
  C_UDMI(c,t,7)=DeltaNW[2]+Lambda*DeltaNA[2];

  if (sqrt(pow(C_UDMI(c,t,2),2)+pow(C_UDMI(c,t,3),2)+pow(C_UDMI(
    c,t,4),2))>1.1)

```

```

{
  Message("Deltat is too large and the magnitude of dn is
    larger than 0.1.\n");
  Message("New director 's length for cell %i is: %f \n",c,sqrt(
    pow(C_UDMI(c,t,5),2)+pow(C_UDMI(c,t,6),2)+pow(C_UDMI(c,t
    ,7),2)));
}

#endif

}
end_c_loop(c,t)
}
}

void Frank()
{
  /*FILE *pf;*/
  Domain *domain=Get_Domain(1);
  Thread *t;
  Thread *tf;
  cell_t c;
  cell_t cn,c_n;
  face_t f;
  int Neighbors;
  int n,i;
  double k=1e-2; /*Franks elastic constant*/
  double Energy=0.0,deltat=1e-4,Rotational_viscosity=10.0;
  double nx,ny,nz,nx_c,ny_c,nz_c,nx_i,ny_i,nz_i;
  double delta_mag,theta_i,deltan_x,deltan_y,deltan_z;
  thread_loop_c(t,domain)
  {
    begin_c_loop(c,t)
    {

```

```

Neighbors=0; /*number of neighbors of a central cell*/ /*we
  will add to the number of neighbors by
    looping over the faces of each cell and
    checking if there is a cell on the other
    side of the face. if a face of the cell is on
    the boundary of the domain, there will be no
    neighboring cell on that face.*/
deltan_x=0.0;
deltan_y=0.0;
deltan_z=0.0;
c_face_loop(c,t,n) /*Loops over all the faces of a cell */
{
  /* n is local face number */
  f=C_FACE(c,t,n); /* returns the global face number using the
    local face number */
  tf=C_FACE_THREAD(c,t,n); /* is used to reference the
    associated face thread */
  if(!BOUNDARY_FACE_THREAD_P(tf)) /*if the face tf is not on
    the boundary of the domain*/
  {
    Neighbors++;
    if(F_C0(f,tf)==c)
    {
#if RP_2D
      /*the neighboring cell's index*/
      c_n=F_C1(f,tf);

      /* x and y components of the center cell's director */
      nx_c=C_UDMI(c,t,1);
      ny_c=C_UDMI(c,t,2);

      /* x and y components of the i-th neighbor of the center
        cell's director */
      nx_i=C_UDMI(c_n,t,1);
      ny_i=C_UDMI(c_n,t,2);

```

```

/*theta_i is the angle between center cell's director and
   neighbor i-th director*/
theta_i=acos((nx_i*nx_c+ny_i*ny_c)/(sqrt(pow(nx_i,2)+pow(
   ny_i,2))*sqrt(pow(nx_c,2)+pow(ny_c,2))+1e-10));

/*the effect of i-th neighbor on the n is calculated
here and then added to the effect of other neighbors*/
delta_mag=deltat*k*sin(theta_i)*cos(theta_i)/
   Rotational_viscosity;
deltan_x+=delta_mag*(nx_i-nx_c);
deltan_y+=delta_mag*(ny_i-ny_c);

Energy+=k/2.0*pow(sin(theta_i),2);

//      Message("Franks Energy is: %f \n", Energy);
#endif

#if RP_3D
/*the neighboring cell's index*/
c_n=F_C1(f,tf);

/* coordinates of the center cell's director */
nx_c=C_UDMI(c,t,2);
ny_c=C_UDMI(c,t,3);
nz_c=C_UDMI(c,t,4);

/* coordinates of the i-th neighboring cell*/
nx_i=C_UDMI(c_n,t,2);
ny_i=C_UDMI(c_n,t,3);
nz_i=C_UDMI(c_n,t,4);

/*theta_i is the angle between center cell's director and
   neighbor i-th director*/

```

```

theta_i=acos((nx_i*nx_c+ny_i*ny_c+nz_i*nz_c)/(sqrt(pow(nx_i
,2)+pow(ny_i,2)+pow(nz_i,2))*sqrt(pow(nx_c,2)+pow(ny_c
,2)+pow(nz_c,2))+1e-10));

/*the effect of i-th neighbor on the n is calculated
here and then is uadded to the effect of other neighbors*/
delta_mag=delta_t*k*sin(theta_i)*cos(theta_i)/
    Rotational_viscosity;
deltan_x+=delta_mag*(nx_i-nx_c);
deltan_y+=delta_mag*(ny_i-ny_c);
deltan_z+=delta_mag*(nz_i-nz_c);

Energy+=k/2.0*pow(sin(theta_i),2);
#endif
}
else
{
#if RP_2D
/*the neighboring cell's index*/
c_n=F_C0(f,tf);

/* x and y components of the center cell's director */
nx_c=C_UDMI(c,t,1);
ny_c=C_UDMI(c,t,2);

/* x and y components of the i-th neighbor of the center
cell's director */
nx_i=C_UDMI(c_n,t,1);
ny_i=C_UDMI(c_n,t,2);

/*theta_i is the angle between center cell's director and
neighbor i-th director*/
theta_i=acos((nx_i*nx_c+ny_i*ny_c)/(sqrt(pow(nx_i,2)+pow(
ny_i,2))*sqrt(pow(nx_c,2)+pow(ny_c,2))+1e-10));

```

```

/*the effect of i-th neighbor on the n is calculated
here and then added to the effect of other neighbors*/
delta_mag=deltat*k*sin(theta_i)*cos(theta_i)/
    Rotational_viscosity;
deltan_x+=delta_mag*(nx_i-nx_c);
deltan_y+=delta_mag*(ny_i-ny_c);

Energy+=k/2.0*pow(sin(theta_i),2);
#endif

#if RP_3D
/*the neighboring cell's index*/
c_n=F_C0(f,tf);

/* director components of the center cell's director */
nx_c=C_UDMI(c,t,2);
ny_c=C_UDMI(c,t,3);
nz_c=C_UDMI(c,t,4);

/* director components of the i-th neighboring cell*/
nx_i=C_UDMI(c_n,t,2);
ny_i=C_UDMI(c_n,t,3);
nz_i=C_UDMI(c_n,t,4);

/*theta_i is the angle between center cell's director and
neighbor i-th director*/
theta_i=acos((nx_i*nx_c+ny_i*ny_c+nz_i*nz_c)/(sqrt(pow(nx_i
,2)+pow(ny_i,2)+pow(nz_i,2))*sqrt(pow(nx_c,2)+pow(ny_c
,2)+pow(nz_c,2))+1e-10));

/*the effect of i-th neighbor on the n is calculated
here and then is uadded to the effect of other neighbors*/
delta_mag=deltat*k*sin(theta_i)*cos(theta_i)/
    Rotational_viscosity;

```



```

    deltan_x+=delta_mag*(nx_i-nx_c);
    deltan_y+=delta_mag*(ny_i-ny_c);
    deltan_z+=delta_mag*(nz_i-nz_c);

    Energy+=k/2.0*pow(sin(theta_i),2);
#endif
    }
    }
}

/*Adding the summed deltan to the components of the director
of the central cell*/
#if RP_2D
    C_UDMI(c,t,3)+=deltan_x;
    C_UDMI(c,t,4)+=deltan_y;
#endif

#if RP_3D
    C_UDMI(c,t,5)+=deltan_x;
    C_UDMI(c,t,6)+=deltan_y;
    C_UDMI(c,t,7)+=deltan_z;
#endif

}
end_c_loop(c,t)
}
// Message(" Total Franks Energy is: %f ", Energy);

// Message(" Last theta is: %f is done.\n", theta_i);

}

void Translation()
{

```

```

/*FILE *pf;*/
Domain *domain=Get_Domain(1);
Thread *t,*tf;
cell_t c,cn,c_n;
face_t f;
int Neighbors,n,i;/**/
double deltat=1e-4,A[ND_ND],ds,es[ND_ND],A_by_es,dr0[ND_ND],dr1[
    ND_ND],vel[ND_ND],Energy=0.0,dx,Angle,alpha;

thread_loop_c(t,domain) /*This part sets the correction UDMs to
    zero before each run*/
{
    begin_c_loop(c,t)
    {
#if RP_2D
        C_UDMI(c,t,3)=0.0; /*the effect of neighbors on nx of cell c*/
        C_UDMI(c,t,4)=0.0; /*the effect of neighbors on ny of cell c*/
        C_UDMI(c,t,5)=0.0; /*number of neighbors contributing to a
            cell c*/
        C_UDMI(c,t,6)=0.0;
#endif
#if RP_3D
        C_UDMI(c,t,5)=0.0; /*the effect of neighbors on nx of cell c*/
        C_UDMI(c,t,6)=0.0; /*the effect of neighbors on ny of cell c*/
        C_UDMI(c,t,7)=0.0; /*the effect of neighbors on nz of cell c*/
        C_UDMI(c,t,8)=0.0; /*number of neighbors contributing to a cell
            */
#endif
    }
    end_c_loop(c,t)
}

thread_loop_c(t,domain) /*This part loops over all cells and
    calculates the effect of each cell on it's neighbors*/
{

```

```

begin_c_loop(c,t)
{
Neighbors=0; /*number of neighbors of a cell*/
c_face_loop(c,t,n) /*Loops over all the faces of a cell */
{
/* n is local face number */
f=C_FACE(c,t,n); /* returns the global face number using the
local face number */
tf=C_FACE_THREAD(c,t,n); /* is used to reference the
associated face thread */
if (!BOUNDARY_FACE_THREAD_P(tf)) /*If the face is not a
boundary face*/
{
Neighbors++; /*If face f is an interior face, cell c has a
neighbor on the other side of the face f, so add 1 to the
number of neighbors*/

INTERIOR_FACE_GEOMETRY(f,tf,A,ds,es,A_by_es,dr0,dr1);
/*for face f on thread tf, returns the following to the
solver:
real A[ND_ND] : the area normal vector (always points from
C0 to C1)
real ds : the distance between the cell centroids
real es[ND_ND] : the unit normal vector in the direction
from cell c0 to c1
real A_by_es : value (A.A)/(A.es)
real dr0[ND_ND] : vector that connects the centroid of c0 to
the face centroid
real dr1[ND_ND] : vector that connects the centroid of c1 to
the face centroid*/

ND.SET(vel[0],vel[1],vel[2],C_U(c,t),C_V(c,t),C_W(c,t));
/*assigns the components of the velocity of the center cell
to the vel[ND_ND]*/

```

```

if (F_C0(f,tf)==c)
{ /*C0 is the center cell here*/

    Angle=acos(NV_DOT(A,vel)/(NV_MAG(A)*NV_MAG(vel)+1e-10));
    /*the angle between the velocity vector of the center cell
    and the area normal vector of the face f*/

    dx=NV_MAG(vel)*cos(Angle)*deltat;
    /*the distance traveled by crystals normal to the face f
    due to the motion of the fluid with velocity vel[ND_ND]*/

    if ((Angle<1.5))//&&(C_UDMI(c_n,t,5)<1.0)
    {
        C_UDMI(c,t,6)=dx/ds;

        c_n=F_C1(f,tf); /*the neighboring cell's index*/

#ifdef RP_2D
        C_UDMI(c_n,t,5)+=1.0;
        if ((dx/ds)>=1.0)
        {
            C_UDMI(c_n,t,3)+=C_UDMI(c,t,1);
            C_UDMI(c_n,t,4)+=C_UDMI(c,t,2);

        }
        else
        {
            C_UDMI(c_n,t,3)+=C_UDMI(c_n,t,1);
            C_UDMI(c_n,t,4)+=C_UDMI(c_n,t,2);
            // alpha=acos((C_UDMI(c,t,1)*C_UDMI(c_n,t,1)+C_UDMI(c,t,
            // 2)*C_UDMI(c_n,t,2))/sqrt(pow(C_UDMI(c,t,1),2)+
            // pow(C_UDMI(c,t,2),2))/sqrt(pow(C_UDMI(c_n,t,1),2)+pow(
            C_UDMI(c_n,t,2),2)));

```

```

// if (alpha < 1.571)
// {
// C_UDMI(c_n , t , 3) += (dx/ds) * (C_UDMI(c , t , 1) - C_UDMI(c_n , t
// , 1)) + C_UDMI(c_n , t , 1) ;
// C_UDMI(c_n , t , 4) += (dx/ds) * (C_UDMI(c , t , 2) - C_UDMI(c_n , t
// , 2)) + C_UDMI(c_n , t , 2) ;
// }
// else
// {
// if (C_UDMI(c , t , 1) < 0.0)
// {
// C_UDMI(c_n , t , 3) += (dx/ds) * ((-1.0) * C_UDMI(c , t , 1) - C_UDMI(c_n , t , 1)) + C_UDMI(c_n , t , 1) ;
// C_UDMI(c_n , t , 4) += (dx/ds) * ((-1.0) * C_UDMI(c , t , 2) - C_UDMI(c_n , t , 2)) + C_UDMI(c_n , t , 2) ;
// }
// else
// {
// C_UDMI(c_n , t , 3) += (dx/ds) * (C_UDMI(c , t , 1) + C_UDMI(c_n , t , 1)) - C_UDMI(c_n , t , 1) ;
// C_UDMI(c_n , t , 4) += (dx/ds) * (C_UDMI(c , t , 2) + C_UDMI(c_n , t , 2)) - C_UDMI(c_n , t , 2) ;
// }
// }
}

if ((dx/ds) > 2.5)
{
  Message("dx/ds is: %e for cell %i. deltat should be
  reduced to reduce the dx for the given velocity.\n",
  dx/ds, c);
}

```

```

#endif

#if RP_3D
    C_UDMI(c_n, t, 8) += 1;
    if ((dx/ds) >= 1.0)
    {
        C_UDMI(c_n, t, 5) += C_UDMI(c, t, 2);
        C_UDMI(c_n, t, 6) += C_UDMI(c, t, 3);
        C_UDMI(c_n, t, 7) += C_UDMI(c, t, 4);
    }
    else
    {
        C_UDMI(c_n, t, 5) += C_UDMI(c_n, t, 2); // (dx/ds) * (C_UDMI(c, t
            , 2) - C_UDMI(c_n, t, 2)) + C_UDMI(c_n, t, 2);
        C_UDMI(c_n, t, 6) += C_UDMI(c_n, t, 3); // (dx/ds) * (C_UDMI(c, t
            , 3) - C_UDMI(c_n, t, 3)) + C_UDMI(c_n, t, 3);
        C_UDMI(c_n, t, 7) += C_UDMI(c_n, t, 4); // (dx/ds) * (C_UDMI(c, t
            , 4) - C_UDMI(c_n, t, 4)) + C_UDMI(c_n, t, 4);
    }
}

#endif
}
}
else
{ /* C1 is the center cell here */

// this is wrong! A = (-1.0) * A;
for (i = 0; i < ND; ++i) A[i] = (-1.0) * A[i];
Angle = acos(NV_DOT(A, vel) / (NV_MAG(A) * NV_MAG(vel) + 1e-10));
/* the angle between the velocity vector of the center cell
and the area normal vector of the face f */

dx = NV_MAG(vel) * cos(Angle) * deltat;

```

```

/*the distance traveled by crystals normal to the face f
due to the motion of the fluid with velocity vel[ND_ND]*/

if ((Angle < 1.1)) // && (C_UDMI(c_n, t, 5) < 1.0)
{

C_UDMI(c, t, 6) = dx/ds;

/*the neighboring cell's index*/
c_n = F_C0(f, tf);
#if RP_2D
C_UDMI(c_n, t, 5) += 1.0;
if ((dx/ds) >= 1.0)
{
C_UDMI(c_n, t, 3) += C_UDMI(c, t, 1);
C_UDMI(c_n, t, 4) += C_UDMI(c, t, 2);

}
else
{
C_UDMI(c_n, t, 3) += C_UDMI(c_n, t, 1);
C_UDMI(c_n, t, 4) += C_UDMI(c_n, t, 2);
// alpha = acos((C_UDMI(c, t, 1) * C_UDMI(c_n, t, 1) + C_UDMI(c, t,
// 2) * C_UDMI(c_n, t, 2)) / sqrt(pow(C_UDMI(c, t, 1), 2) +
// pow(C_UDMI(c, t, 2), 2)) / sqrt(pow(C_UDMI(c_n, t, 1), 2) + pow(
// C_UDMI(c_n, t, 2), 2)));
// if (alpha < 1.571)
// {
// C_UDMI(c_n, t, 3) += (dx/ds) * (C_UDMI(c, t, 1) - C_UDMI(c_n, t
// , 1)) + C_UDMI(c_n, t, 1);
// C_UDMI(c_n, t, 4) += (dx/ds) * (C_UDMI(c, t, 2) - C_UDMI(c_n, t
// , 2)) + C_UDMI(c_n, t, 2);
// }
// else
// {

```

```

// if (C_UDMI(c,t,1) < 0.0)
// {
// C_UDMI(c_n,t,3) += (dx/ds) * ((-1.0)*C_UDMI(c,t,1) - C_UDMI(
//   c_n,t,1)) + C_UDMI(c_n,t,1);
// C_UDMI(c_n,t,4) += (dx/ds) * ((-1.0)*C_UDMI(c,t,2) - C_UDMI(
//   c_n,t,2)) + C_UDMI(c_n,t,2);
// }
// else
// {
// C_UDMI(c_n,t,3) += (dx/ds) * (C_UDMI(c,t,1) + C_UDMI(c_n,t
//   ,1)) - C_UDMI(c_n,t,1);
// C_UDMI(c_n,t,4) += (dx/ds) * (C_UDMI(c,t,2) + C_UDMI(c_n,t
//   ,2)) - C_UDMI(c_n,t,2);
// }
// }
}

```

```

if ((dx/ds) > 2.5)
{
  Message("dx/ds is: %e for cell %i. deltat should be
    reduced to reduce the dx for the given velocity.\n",
    dx/ds, c);
}

```

```

// if ((c % 200) == 0) /*min<(DeltaNW[0]/DeltaNW[1])*/
// {
// Message("dx/ds is: %e .\n", dx/ds);
// }

```

```

#endif

```

```

#if RP3D

```

```

  C_UDMI(c_n,t,8) += 1.0; /*number of neighbors*/
  if ((dx/ds) >= 1.0)

```



```

    {
        C_UDMI(c_n , t , 5) += C_UDMI(c , t , 2) ;
        C_UDMI(c_n , t , 6) += C_UDMI(c , t , 3) ;
        C_UDMI(c_n , t , 7) += C_UDMI(c , t , 4) ;

    }
    else
    {
        C_UDMI(c_n , t , 5) += C_UDMI(c_n , t , 2) ; // ( dx/ds ) * ( C_UDMI(c , t , 2)
            - C_UDMI(c_n , t , 2) ) + C_UDMI(c_n , t , 2) ;
        C_UDMI(c_n , t , 6) += C_UDMI(c_n , t , 3) ; // ( dx/ds ) * ( C_UDMI(c , t , 3)
            - C_UDMI(c_n , t , 3) ) + C_UDMI(c_n , t , 3) ;
        C_UDMI(c_n , t , 7) += C_UDMI(c_n , t , 4) ; // ( dx/ds ) * ( C_UDMI(c , t , 4)
            - C_UDMI(c_n , t , 4) ) + C_UDMI(c_n , t , 4) ;
    }
    if ((dx/ds) > 2.5)
    {
        Message("dx/ds is: %e for cell %i. deltat should be
            reduced to reduce the dx for the given velocity.\n",
            dx/ds , c) ;
    }
#endif
    }
    }
    }
    }
    //Message("dx/ds is: %e for cell %i. \n ", dx/ds , c) ;
    }
    end_c_loop(c , t)
    }
    //Message("dx/ds is: %e for cell %i. ", dx/ds , c) ;
}

void Initial_condition ()
{

```

```

Domain *domain=Get_Domain(1);
Thread *t;
cell_t c;
face_t f;
real x[ND_ND];
int i;
thread_loop_c(t, domain)
{
    begin_c_loop(c, t)
    {
        C_CENTROID(x, c, t);

#ifdef RP_2D
        C_UDMI(c, t, 0) = 0.0; // 3.1416 / 2.0;
        Set_User_Memory_Name(0, "angle_from_x_axis");
        Set_User_Memory_Name(1, "directors_x_component");
        Set_User_Memory_Name(2, "directors_y_component");
        C_UDMI(c, t, 1) = cos(C_UDMI(c, t, 0)); /* x component of the director
        */
        C_UDMI(c, t, 2) = sin(C_UDMI(c, t, 0)); /* y component of the director
        */
#endif

#ifdef RP_3D

        if (x[2] < 0.01)
        {
            C_UDMI(c, t, 2) = 1.0;
            C_UDMI(c, t, 3) = 0.0;
            C_UDMI(c, t, 4) = 0.0;
        }
        else
        {
            C_UDMI(c, t, 2) = 0.0;
            C_UDMI(c, t, 3) = 0.0;

```

```

        C_UDMI(c,t,4)=1.0;
    }

#endif

    }
    end_c_loop(c,t)
}
}

void OrderParameter()
{
    Domain *domain=Get_Domain(1);
    Thread *t;
    cell_t c;
    face_t f;
    // real x[ND_ND];
    int i=1,j=1,k;
    real Pi=3.1416;
    real deltatheta=Pi/180.0;
    real theta=0.0,S=0.0;
    real ff[360];
    for(j=1;j<360;j++)
        ff[j]=0.0;

    thread_loop_c(t,domain) /*This part calculates the angle of the
        directors for
        each cell based on the n_x, x_y and in case of 3D simulations
        n_z*/
    {
        begin_c_loop(c,t)
        {
            // C_CENTROID(x,c,t);

#endif RP_2D

```

```

        if (C_UDMI(c, t, 1) == 0) C_UDMI(c, t, 1) = 0.00001;
        C_UDMI(c, t, 0) = atan(C_UDMI(c, t, 2) / C_UDMI(c, t, 1));
    #endif

    #if RP_3D
    #endif
    }
    end_c_loop(c, t)
}

thread_loop_c(t, domain)
{
    begin_c_loop(c, t)
    {
//      C_CENTROID(x, c, t);

    #if RP_2D
    //  i=1;
    for (theta = 0.0; theta <= Pi; theta = theta + deltatheta)
    {
        if (C_UDMI(c, t, 0) >= theta && C_UDMI(c, t, 0) < (theta + deltatheta))
        {
            k = theta * 180 / Pi;
            ff[k+1] = ff[k+1] + 1;
            i++;
            break;
        }
    }
    #endif

    #endif

    #if RP_3D
    /*In the 2D case, x direction is considered as the director
    direction

```

and the integration is done from 0 (x-direction) to Pi (-x direction)  
 In this case, the first UDM is the angle that was calculated in the  
 random() fuction. for the cases in which the effects of the Franks,  
 evolution, and translation is applied, the angle should be calculated  
 from the final orientation stored in C\_UDMI(c,t,1)=n\_x and C\_UDMI(c,t,2)=n\_y

For the 3D case, the director should be given if we want to calculate

the order parameter. \*/

```
#endif
```

```
}
```

```
end_c_loop(c,t)
```

```
}
```

```
// Message("The number of cells are: %d \n", i);
```

```
for(theta=0.0;theta<=Pi;theta=theta+deltatheta)
```

```
{
```

```
k=theta*180/Pi;
```

```
ff[k+1]=ff[k+1]/i;
```

```
}
```

```
// thread_loop_c(t, domain)
```

```
// {
```

```
// begin_c_loop(c,t)
```

```
// {
```

```
#if RP_2D
```

```
j=1;
```

```
for(theta=0.0;theta<=Pi;theta=theta+deltatheta)
```

```
{
```

```
k=theta*180/Pi;
```

```

    S=S+(1.0-3.0/2.0*pow(sin(theta),2))*ff[k+1]; //order parameter
        , surface integral over a unit sphere
    j=j+1;
//    Message(" i= %d, k= %d , j= %d , S= %e , P(j)= %e \n",i , k,
    j , S, ff[k+1]);
    }
#endif

#if RP_3D

#endif
// }
// end_c_loop(c,t)
// }
    Message("The order parameter (S) is: %e, theta is: %e maximum k
        is: %d\n", S, theta , k);
}

```