

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Numerical studies of purely elastic instabilities in micro-geometries: non-homogeneous concentration effects

Francisco Manuel Esteves Cerqueira Alves Martins



Mestrado Integrado em Engenharia Mecânica

Supervisors: Prof. Fernando Manuel Coutinho Tavares de Pinho and Prof. Alexandre Miguel Prior Afonso

June 28, 2021

**Numerical studies of purely elastic instabilities in
micro-geometries: non-homogeneous concentration
effects**

Francisco Manuel Esteves Cerqueira Alves Martins

Mestrado Integrado em Engenharia Mecânica

June 28, 2021

Abstract

Flow stability is a challenging topic of research and is a relevant engineering issue. Flow instabilities take place in many different ways and situations, depending on fluid type, flow velocity and geometry in addition to the simultaneous presence of other physical phenomena (such as temperature-dependent density, concentration or surface tension gradients, amongst others).

This Master thesis focus on purely elastic instabilities of multiphase flows in a cross-slot micro geometry, i.e. instabilities in creeping flows of viscoelastic fluids, using numerical methods. For these numerical studies, the opensource code OpenFOAM (Open Field Operation And Manipulation) is selected, because it is widely accepted by the computational fluid dynamics community, both in academia and industry. We use RheoTool [16], an open-source toolbox based on OpenFOAM to simulate multiphase viscoelastic fluid flows, namely the *rheoInterFoam* solver.

Previous investigations on cross-slot flow were carried out assuming single-phase fluids and here we extend them assuming non-homogeneous inlet configurations, using two different non miscible fluids at each cross-slot inlet. These fluids are defined by multiphase viscoelastic models, such as the Upper-Convected Maxwell and the Oldroyd-B models, at different elasticity magnitudes, defined by the Weissenberg or Deborah numbers. The main objective of this work is to study the effects of the different flow and multiphase fluid configurations on the onset of flow instabilities.

In a Newtonian/UCM flow several studies are presented, regarding the creeping flow setting, the capillary number influence and the Reynolds number ratio between the fluids. An evaluation of the UCM/UCM flow is also presented, where the effects of different Deborah number pairs are evaluated for a flow with a low Reynolds number and no surface tension. Other flow properties such as the stress field, the velocity profile, the stagnation point position and the interface representation are studied in order to better understand the ongoing physics of these flows. Finally a small analysis of the Oldroyd-B/Oldroyd-B fluids is presented, evaluating the effects of the addition of solvent viscosity in multiphase viscoelastic flows. For all simulated flows, the flow state and local Weissenberg number in the steady cases are showcased.

Acknowledgements

I will take this opportunity to thank my supervisors, Professor Fernando Pinho and Professor Alexandre Afonso, for the continuous and ceaseless technical and moral support they gave me throughout this journey, for all the resources they made available for me to use when I saw mine fleeing away and for the valuable knowledge and experience they shared with me.

I would also like to thank my friends for the aid and knowledge they provided me in the last few months, for the needed distractions but also work ethics they allow me to fully engage in, and for always being there with me in the ups and downs we call living.

And lastly I thank my family for all the love and care I will be thanking my whole life, and for giving me all the tools one could possibly need to invent his own happiness.

Francisco Martins

*“Está provado,
quem espera nunca alcança.”*

Chico Buarque

Contents

1	Introduction	1
2	Bibliographic Revision	3
2.1	Viscoelastic fluids	3
2.2	Stagnation point in the cross-slot geometry	5
2.3	Newtonian fluid flow in a cross-slot setting	5
2.4	Viscoelastic fluid flow in a cross-slot setting	7
2.5	Multiphase flows	9
3	Fundamental Equations and Numerical Method	11
3.1	Introduction	11
3.2	Governing equations	12
3.3	Constitutive equations	13
3.4	Constitutive models for viscoelastic fluid flows	13
3.4.1	Upper-convected Maxwell model	13
3.4.2	Oldroyd-B model	15
3.5	Numerical method	16
3.6	Numerical stabilization for viscoelastic fluid flows simulations	17
3.6.1	The log-conformation tensor approach	17
3.6.2	Pressure-velocity coupling algorithm	19
3.6.3	Stress-velocity coupling	19
3.6.4	High-resolution schemes	20
3.6.5	Log-conformation application to the UCM and Oldroyd-B models	21
3.6.6	Multiphase specifications	21
4	Software Description	23
4.1	Folder organization	23
4.2	Pre-processing	24
4.2.1	Mesh criation	24
4.2.2	Boundary conditions	25
4.2.3	Discretized equations and solution	25
4.2.4	Simulation control	26
4.3	Starting a simulation	26
4.4	Post-processing	27
4.4.1	Flow steadiness	27
4.4.2	Velocity profile	28
4.4.3	Interface representation	28
4.4.4	Local Weissenberg calculation	29

4.4.5	Stress plotting	30
4.5	Final considerations	30
5	Mesh dependency study	31
5.1	Mesh characteristics	31
5.2	Mesh Analysis	34
5.3	Mesh type comparison	35
5.4	Conclusions	35
6	Interfacial flow between UCM and Newtonian fluids	37
6.1	Introduction	37
6.2	Inertialess flow	37
6.2.1	Simulation parameters	37
6.2.2	Results	38
6.3	Unitary Reynolds flow	38
6.3.1	Simulations performed	38
6.4	Capillary number effects	39
6.5	Reynolds ratio consideration	42
6.5.1	Unitary Reynolds for the Newtonian fluid flow	42
6.5.2	Unitary Reynolds for the UCM model fluid flow	46
6.5.3	Steady asymmetric flow mapping	51
7	Interfacial flow between multiphase UCM fluids	55
7.1	Introduction	55
7.2	Flow description	55
7.3	Results	56
7.4	Near Symmetry breaking flows study	59
8	Interfacial flow between multiphase Oldroyd-B fluids	65
8.1	Introduction	65
8.2	Flow description	65
8.3	Results	66
9	Conclusions	73
A	RheoInterFoam Files	77
A.1	blockMeshDict	78
A.2	boundary	81
A.3	constitutiveProperties	82
A.4	controlDict	83
A.5	fvSchemes	86
A.6	fvSolution	88
A.7	decomposeParDict	89
A.8	setFields	90
B	Post-processing code	91
C	Powerpoint presentation	107
	References	117

List of Figures

1.1	Schematic of cross-slot geometry.	2
2.1	Distribution of velocity magnitude at the entrance region of the outlet arms (along the lines $y = \pm 0.5$) for different meshes for Newtonian creeping flow.	6
2.2	Distribution of the flow-type parameter and the non-dimensional xy component of the vorticity tensor (Ω_{xy}) for a Newtonian fluid with superimposed streamlines.	7
2.3	Flow patterns in the cross-slot geometry for a PAA300+50%Glyc viscoelastic fluid: (i) Newtonian-like symmetric flow behaviour, (ii) steady asymmetric flow, (iii) time-dependent flow at a given instant.	7
2.4	$Re - De$ map for the Oldroyd-B viscoelastic fluid flow with $\beta = 1/9$	8
2.5	Optical micrographs of the microchannel (a) with and (b) without cavity. The anticipated location of the stagnation point (S.P.) sitting on a free streamline (a) or pinned onto the confining wall (b) is indicated in the channel. (a) Microchannel with cavity. (b) Microchannel without cavity.	9
2.6	3D view of droplet spreading time evolution by Figueiredo et al.	10
3.1	Graphical representation of the stress tensor in a Cartesian referential.	12
3.2	Mechanical model equivalent to a Maxwell fluid.	14
3.3	Mechanical model equivalent to the Oldroyd-B model.	15
3.4	Control volume and faces orientation. W (west), E (east), S (south), N (north), B (bottom), T (top).	17
4.1	Directory structure of a general OpenFOAM case.	24
4.2	Example of the U_x/U_{mag} evolution over time.	27
4.3	Interface representation for an asymmetric flow. The square represents the position of the stagnation point.	28
4.4	Variable illustration of the lambda calculation. The square represents the stagnation point location.	29
5.1	Schematics of the cross-slot geometry.	32
5.2	Schematics of the cross-slot unrefined geometry without cell size gradient.	32
5.3	Schematics of the cross-slot unrefined geometry with cell size gradient and respective blocks used to generate the mesh.	33
6.1	Ca-De phase diagram for UCM/Newtonian flows at $Re=1$	40
6.2	Flow state for each Reynolds number ratio, as a function of the Deborah number, with a fixed $Re_N = 1$ and $Ca = \infty$	43

6.3	Interface representation and respective stagnation point coordinates (s.p.) when $De=0.1$, $Re_N = 1$ and $Re_{UCM}/Re_N =$: a) 0.01, s.p.: $(x,y^*)=(0.00,0.04)$; b) 0.1, s.p.: $(x,y^*)=(0.00,0.03)$; c) 0.5, s.p.: $(x,y^*)=(0.00,0.03)$; d) 1, s.p.: $(x,y^*)=(0.00,0.03)$; e) 2, s.p.: $(x,y^*)=(0.00,0.02)$; f) 10, s.p.: $(x,y^*)=(0.00,-0.05)$; g) 100, s.p.: $(x,y^*)=(0.00,-0.30)$. The square represents the stagnation point above which is the UCM fluid and below it, the Newtonian.	44
6.4	Flow state for each Reynolds ratio, as a function of the Deborah number, with a fixed $Re_{UCM} = 1$ and $Ca = \infty$	47
6.5	Interface representation and respective stagnation point coordinates (s.p.) when $De=0.1$, $Re_{UCM} = 1$ and $Re_{UCM}/Re_N =$: a) 0.01, s.p.: $(x,y^*)=(-0.16,0.37)$; b) 0.1, s.p.: $(x,y^*)=(0.00,0.10)$; c) 0.5, s.p.: $(x,y^*)=(0.00,0.04)$; d) 1, s.p.: $(x,y^*)=(0.00,0.03)$; e) 2, s.p.: $(x,y^*)=(0.00,0.02)$; f) 10, s.p.: $(x,y^*)=(0.00,0.02)$; g) 100, s.p.: $(x,y^*)=(0.00,0.20)$. The square represents the stagnation point above which is the UCM fluid and below it, the Newtonian.	48
6.6	Normalized velocity profiles in absolute values for the cross-slot arms inlets at $x=0.5$ and $x=-0.5$, for the previous a), d) and g) cases. h represents the channels width, U represents the inlet average velocity of the flow.	49
6.7	$De - Re_{UCM}/Re_N$ phase diagrams in the range 0.018-0.056 for $Re_{UCM} = 1$ and $Ca = \infty$	52
7.1	$De_1 - De_2$ phase diagram for UCM/UCM flow at $Re_1 = 0.01$, $Re_2 = 0.01$ and $Ca = \infty$	57
7.2	Stagnation point normalized distance from the geometry center, of every simulated steady asymmetric flow according to the $ De_{UCM_1} - De_{UCM_2} $ difference.	59
7.3	Interface representation of the near critical and critical flow conditions. a) $De_{UCM_2} = 0.29$, b) $De_{UCM_2} = 0.30$, c) $De_{UCM_2} = 0.31$, d) $De_{UCM_2} = 0.32$, e) $De_{UCM_2} = 0.33$, f) $De_{UCM_2} = 0.34$	61
7.4	Velocity profile representation, for a fixed Deborah of 0.32 in the upper flow.	62
7.5	Polymeric extra-stress tensor magnitude in the center region of the cross-slot geometry. a) $De_{UCM_2} = 0.29$, b) $De_{UCM_2} = 0.30$, c) $De_{UCM_2} = 0.31$, d) $De_{UCM_2} = 0.32$, e) $De_{UCM_2} = 0.33$, f) $De_{UCM_2} = 0.34$	63
8.1	Interface representation for three flows with $\beta_1 = 0.1$, $De_1 = 0.33$, $De_2 = 0.33$, $Re=0.01$ and $Ca=\infty$. a) $\beta_2 = 0.1$; b) $\beta_2 = 0.5$; c) $\beta_2 = 0.9$	71
8.2	Plotting of the local Weissenberg, β ratio and stagnation point displacement relation for the flows with $De_1 = 0.29$, $Re=0.01$ and $Ca=\infty$. H stands for channel width; $Dist$ stands for stagnation point distance from the geometry center.	72

List of Tables

5.1	Characteristics of the computational meshes. NP - number of points; NC - number of cells; D - channel width	34
5.2	UCM model data for $De=0.330$. Wi_0 - local Weissenberg number; DQ - flow asymmetry parameter; t - order of simulation time - t; ϵ_1 - local Weissenberg relative error; ϵ_2 - asymmetry parameter relative error;	34
5.3	UCM model/Newtonian data for $Re=1$ and in absence of surface tension.	35
6.1	Critical Deborah number (De) and local Weissenberg number (Wi_0) values. . . .	38
6.2	UCM model/Newtonian data for $Re = 1$ and $Ca = \infty$. * stands for unsteady flow. . . .	39
6.3	Local Weissenberg numbers for every simulated De and Ca pair. * represents unsteady flows, - represents not simulated pairs.	41
6.4	Local Weissenberg numbers for every simulated De and Reynolds number ratio, with $Re_N = 1$. * represents Unsteady flows, - represents not simulated pairs, ' represents steady asymmetry	45
6.5	Local Weissenberg numbers for every simulated De and Reynolds ratio, with $Re_{UCM} = 1$. * represents Unsteady flows, - represents not simulated pairs, ' represents steady asymmetry	50
6.6	Local Weissenberg numbers for every simulated De and Reynolds number ratio, with $Re_{UCM} = 1$. * represents Unsteady flows.	52
7.1	Local Weissenberg values for all simulated quantities in this chapter. * stands for unsteady flow, ' stands for steady asymmetric flow.	58
7.2	Stagnation point stress stress magnitude contributions. $De_1 = 0.32$, $Re=0.01$, $Ca = \infty$ and the stagnation point is located at the center of the geometry ((x,y)=(0,0)). . .	63
8.1	Local Weissenberg values for all simulated flows with $De_1 = 0.29$, with $Ca=\infty$ and $Re=0.01$. Dist. stands for the stagnation point geometry center distance.	66
8.2	Local Weissenberg values for all simulated flows with $De_1 = 0.30$, with $Ca=\infty$ and $Re=0.01$. Dist. stands for the stagnation point geometry center distance.	67
8.3	Local Weissenberg values for all simulated flows with $De_1 = 0.31$, with $Ca=\infty$ and $Re=0.01$. * represents unsteady flows, Dist. stands for the stagnation point geometry center distance.	68
8.4	Local Weissenberg values for all simulated flows with $De_1 = 0.32$, with $Ca=\infty$ and $Re=0.01$. * represents unsteady flows, ' stands for steady asymmetric flow, Dist. stands for the stagnation point geometry center distance.	69
8.5	Local Weissenberg values for all simulated flows with $De_1 = 0.33$, with $Ca=\infty$ and $Re=0.01$. * represents unsteady flows, ' stands for steady asymmetric flow, Dist. stands for the stagnation point geometry center distance.	70

Abbreviations

FEUP	Faculdade de Engenharia da Universidade do Porto
CEFT	Centro de Estudos de Fenómenos de Transporte
FVM	Finite Volumes Method
CFD	Computational Fluid Dynamics
UCM	Upper-convected Maxwell model
CUBISTA	Convergent and Universally Bounded Interpolation Scheme for Treatment of Advection
SIMPLEC	Semi-Implicit Method for Pressure Linked Equations-Consistent
HRS	High-Resolution Schemes

Chapter 1

Introduction

The growing interest in viscoelastic fluid flows and their unique nature, allied with the necessity of investigation for the cross-slot geometrical configuration are some of the motivations for the development of this numerical study. This dissertation contains the study of both single-phase and two-phase fluid flows, described by the UCM and Oldroyd-B constitutive models, in addition to their interaction with Newtonian fluid flows. The aid of numerical tools enable detailed observations on the grounds of flow steadiness and its limits. Geometry wise, the cross-slot configuration is the selected setting for this thesis, as it is a candidate for a numerical benchmark flow problem for viscoelastic fluids [6]. The cross-slot geometry, shown in Figure 1.1 is formed by the intersection between two long straight planar channels where the flow comes in through two opposed channels and leaves through the other two channels. For single-phase fluids the orientation of the incoming and outgoing channels is irrelevant, but for multiphase fluids the orientation to gravity may be of relevance [20]. This flow geometry has been extensively utilized for generating controllable planar, uniaxial and biaxial elongation flow fields and to study the stretching dynamics of polymers [23]. Since this dissertation is mainly focused in the study of flow effects of extensional nature and the corresponding extensional behaviour of the fluids, the cross-slot configuration is selected as it minimizes effects of shear.

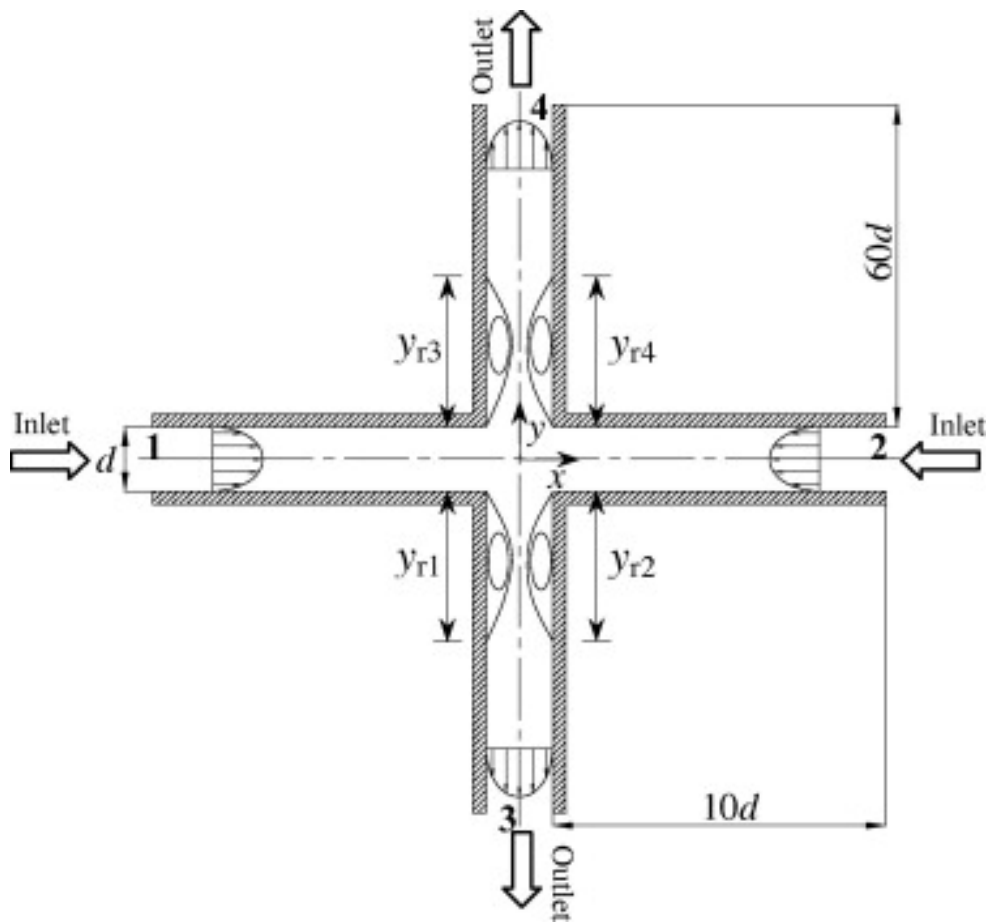


Figure 1.1: Schematic of cross-slot geometry [20].

All numerical simulations performed, use the OpenFOAM software through the rheoTool toolbox, developed by Pimenta and Alves [16].

Chapter 2

Bibliographic Revision

2.1 Viscoelastic fluids

The description of viscoelastic fluids, requires a previous differentiation between Newtonian and non-Newtonian fluids.

In simple shear flows, like the Couette flow, Newtonian fluids like water, follow a law that defines a linear relation between shear stresses, τ_{xy} , and deformation rates, $\dot{\gamma}$,

$$\tau_{xy} = \eta \dot{\gamma} \quad (2.1)$$

where the viscosity, η is constant and represents the proportionality coefficient. In non-Newtonian fluids this isn't necessarily the case and the viscosity varies with the deformation rate (Equation 2.2).

$$\eta(\dot{\gamma}) = \frac{\tau_{xy}}{\dot{\gamma}} \quad (2.2)$$

Another relevant characteristic of these fluids is the normal stress introduction in simple shear flows (τ_{xx} , τ_{yy} and τ_{zz}), which does not happen in analogous Newtonian fluid flows. This is associated with the several degrees of freedom enabled by the molecular structure of non-Newtonian fluids. [2].

In the non-Newtonian fluids domain, we can find fluids presenting higher viscosity with the increase of the deformation rate, (water and corn starch mixture, moving sands), fluids whose viscosity decreases with the deformation rate increase (yogurt, paint) and also fluids that react to applied forces with viscous deformations (of dissipative nature), and elastic deformations (reversible). Polymer solutions are examples of viscoelastic fluids.

Due to this elastic behaviour, these fluids have the capacity to store some energy, besides the energy they dissipate, giving them a memory-like effect. Two exclusive characteristics of viscoelastic fluids are the retardation time, or elasticity delay and the relaxation time. Applying stress to a fluid with elastic properties, involves a delay in its the response time due to the fluids storing energy capacity (retardation), unlike viscous fluids that show an instantaneous reaction. When stress is no longer applied, the fluid tends to slowly recover part of its initial state, meaning

that the time the fluid takes to recover its initial conditions is of the order of the relaxation time. In sum, a viscoelastic fluid flow does not only depend on the current fluid conditions, but is also influenced by the history of its deformation [2].

There are several constitutive equations to describe this viscoelastic behavior. In this work the models developed by Maxwell [13] and Oldroyd [14] are the ones addressed. These governing equations will be introduced further ahead in this dissertation.

In terms of the main dimensionless numbers used to describe viscoelastic fluids, we characterize the Reynolds number (Re), the Weissenberg number (Wi), the Deborah number (De) and the Capillary number (Ca).

The Reynolds number relates inertial and viscous effects and its use is very popular in fluid mechanics as it predicts flow behaviour, namely the transition from a laminar regime to a turbulent regime. This quantity can be calculated with the following expression

$$Re = \frac{\rho H U}{\eta} \quad (2.3)$$

where ρ is the specific mass of the fluid, H is a characteristic length, U is a characteristic velocity and η represents the fluids viscosity.

The elasticity of a fluid can be characterized by the Weissenberg number (Wi). This quantity is obtained by the product of the fluids relaxation time (λ) with a characteristic deformation rate ($\dot{\gamma}$) as observed in Equation (2.4). Since Wi is a dimensionless number, the characteristic deformation rate is defined, as the ratio between the flows characteristic velocity (U) and its characteristic length (H),

$$Wi = \lambda \dot{\gamma} = \lambda \frac{U}{H}. \quad (2.4)$$

The Deborah number is often used in rheology to characterize the fluidity of materials, under specific conditions. It quantifies the observation, that given enough time even a solid-like material might flow, or a fluid-like material can act solid when is deformed rapidly enough. The Deborah number follows the equation

$$De = \frac{t_c}{t_p} \quad (2.5)$$

where t_c symbolizes the relaxation time of the fluid, and t_p characterizes the "time of observation", or more commonly, the time scale of the process.

The Deborah and Weissenberg numbers are often confused in technical literature, and have different physical interpretations. Physically, the Deborah number represents the ratio of a characteristic time of the fluid (the relaxation time) to that of the flow, and the Weissenberg number represents the ratio of elastic to viscous stresses in the flow [19]. Despite the differences De and Wi are identical and used interchangeably very often for a given flow.

The last important dimensionless number utilized in the body of work is the Capillary number. This quantity represents the relative effect of viscous drag forces versus surface tension forces

acting across the interface of two immiscible fluids. Its definition follows

$$Ca = \frac{\mu V}{\sigma} \quad (2.6)$$

where μ is the dynamic viscosity of the fluid, V is a characteristic velocity of the flow and σ represents the surface tension between the two phases.

2.2 Stagnation point in the cross-slot geometry

In the cross-slot geometry the opposing inlets and outlets produce a flow field with a free stagnation point located in the centre region of the geometry. At this point the velocity field is null and a fluid element is trapped for an infinite time, generating a significant strain and potentially enabling steady-state extensional flow kinematics to be done which is why this geometry is proposed as an extensional rheometer [23]. In terms of simulation analysis, the stagnation point vicinity ends up being the most interesting region of study, as its location and local Weissenberg number are good comparison quantities between different flows. The local Weissenberg number, $Wi_0 = \lambda \dot{\epsilon}_0$ is calculated using the strain rate $\dot{\epsilon}_0$ at the stagnation point.

$$\dot{\epsilon}_0 = \sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \frac{\partial u}{\partial y} \cdot \frac{\partial v}{\partial x}} \quad (2.7)$$

In the proximity of the stagnation point the velocity field can be generally considered linear, and should be approximated by $[u, v] = [ax + by, cx - ay]$, with $a = (\partial u / \partial x) = -(\partial v / \partial y)$, $b = (\partial u / \partial y)$ and $c = (\partial v / \partial x)$. In matricial form the velocity field is written as $\mathbf{u} = \mathbf{A}\mathbf{x}$. The positive eigenvalue of this linear transformation provides the extensional strain rate given by Eq.(2.7), and the principal axes defined by the streamlines that cross the stagnation point are represented by the corresponding eigenvectors. Note that these streamlines are not normal to each other in a steady asymmetric flow. For symmetric flow the strain rate at the stagnation point is given by $\dot{\epsilon}_0 = -(\partial u / \partial x) = (\partial v / \partial y)$ [6].

2.3 Newtonian fluid flow in a cross-slot setting

The Newtonian fluid flow in a cross-slot setting is an extensively studied case for the geometry at hand since its simplicity almost represents the characteristics of the geometry itself. From the many Newtonian creeping flow studies available in the literature, the work done by M. Davoodi et al. [7] presents a suitable introduction to this matter. In the mentioned work [7], the flow distribution of velocity magnitude at the entrance region of the outlet arms is presented for two mesh refinements, and it can be visualized in the following figure, providing a notion of fluid behaviour in this geometry.

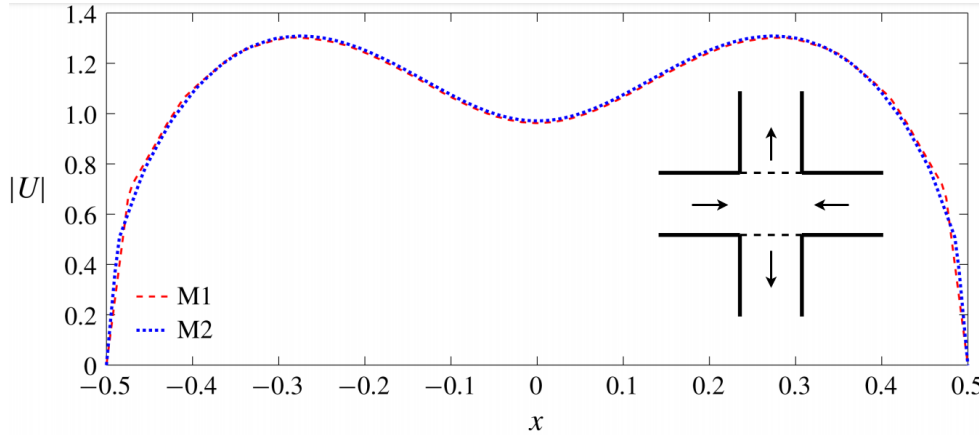


Figure 2.1: Distribution of velocity magnitude at the entrance region of the outlet arms (along the lines $y = \pm 0.5$) for different meshes for Newtonian creeping flow [7].

Another flow behaviour insight provided in [7] is the study of the flow parameter ξ [12]:

$$\xi = \frac{\|\tilde{\mathbf{D}}\| - \|\tilde{\mathbf{\Omega}}\|}{\|\tilde{\mathbf{D}}\| + \|\tilde{\mathbf{\Omega}}\|} \quad (2.8)$$

where $\|\tilde{\mathbf{D}}\| = ((\tilde{\mathbf{D}} : \tilde{\mathbf{D}})/2)^{1/2}$ and $\|\tilde{\mathbf{\Omega}}\| = ((\tilde{\mathbf{\Omega}} : \tilde{\mathbf{\Omega}})/2)^{1/2}$ are the magnitudes of the rate of deformation (i.e. $\tilde{\mathbf{D}} = (\nabla \tilde{\mathbf{U}} + \nabla \tilde{\mathbf{U}}^T)/2$) and vorticity tensors (i.e. $\tilde{\mathbf{\Omega}} = (\nabla \tilde{\mathbf{U}} - \nabla \tilde{\mathbf{U}}^T)/2$), respectively. The ξ parameter may vary within the range $[-1, 1]$ in which $\xi = -1$ characterizes a solid-like rotational flow, $\xi = 1$ a pure extensional flow and $\xi = 0$ a simple shear flow. As shown in Figure 2.2, in the creeping flow of Newtonian fluids, the extensional-dominated flow appears in the shape of four strands stretched along the centerlines of the arms. These strands are located at the mid-distance between the two walls where, due to symmetry, the shear rate is zero, and due to a non-zero streamwise gradient of velocity, a purely elongational flow is observed. This effect is also highlighted by plotting the xy component of the vorticity tensor (Ω_{xy}) as shown in the right-hand column of Figure 2.2.

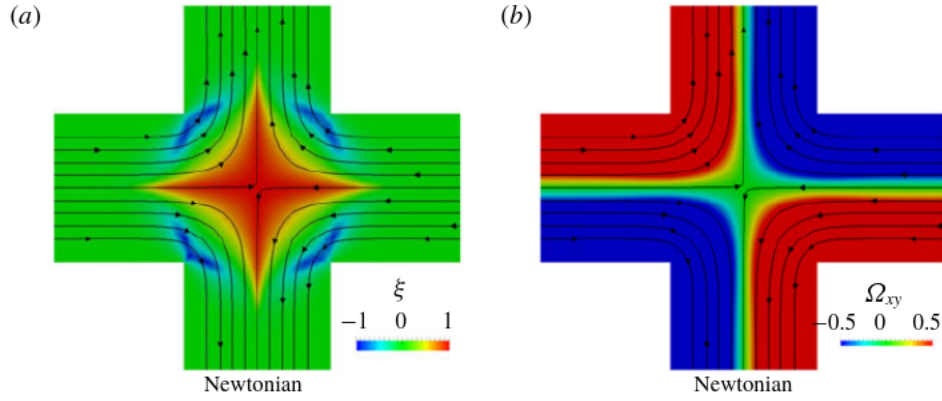


Figure 2.2: Distribution of the flow-type parameter and the non-dimensional xy component of the vorticity tensor (Ω_{xy}) for a Newtonian fluid with superimposed streamlines [7].

2.4 Viscoelastic fluid flow in a cross-slot setting

The single-phase viscoelastic fluid flow in a cross-slot setting is an extensively studied case in the literature and the documented flow patterns are an important subject this dissertation addresses. The work of Sousa et al. [26] gives a good presentation on what these patterns might be, by investigating the effects of polymer solution concentration and the microchannel bounding walls under negligible inertial conditions. It was demonstrated that above a first critical Weissenberg number value the flow becomes asymmetric while remaining steady and above a second critical Weissenberg number the flow becomes unsteady and fluctuates non-periodically in time (see Figure 2.3).

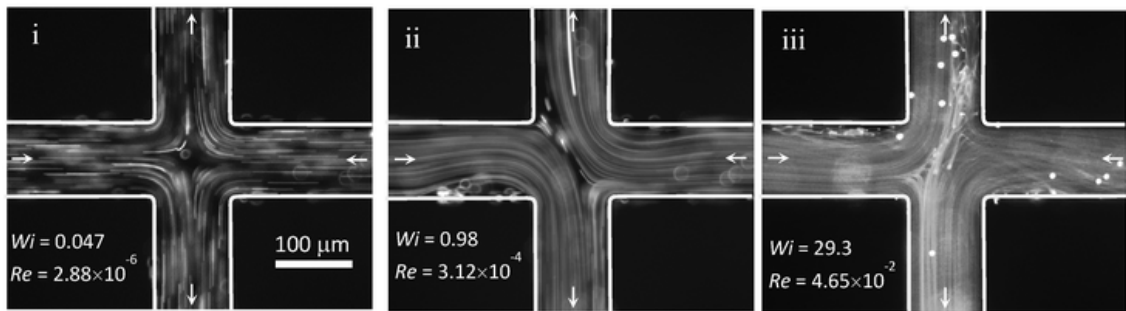


Figure 2.3: Flow patterns in the cross-slot geometry for a PAA300+50%Glyc viscoelastic fluid: (i) Newtonian-like symmetric flow behaviour, (ii) steady asymmetric flow, (iii) time-dependent flow at a given instant [26].

Inertia effects on flow patterns were also studied by Poole et al. [18] and the resulting flow pattern mapping can be observed in Figure 2.4.

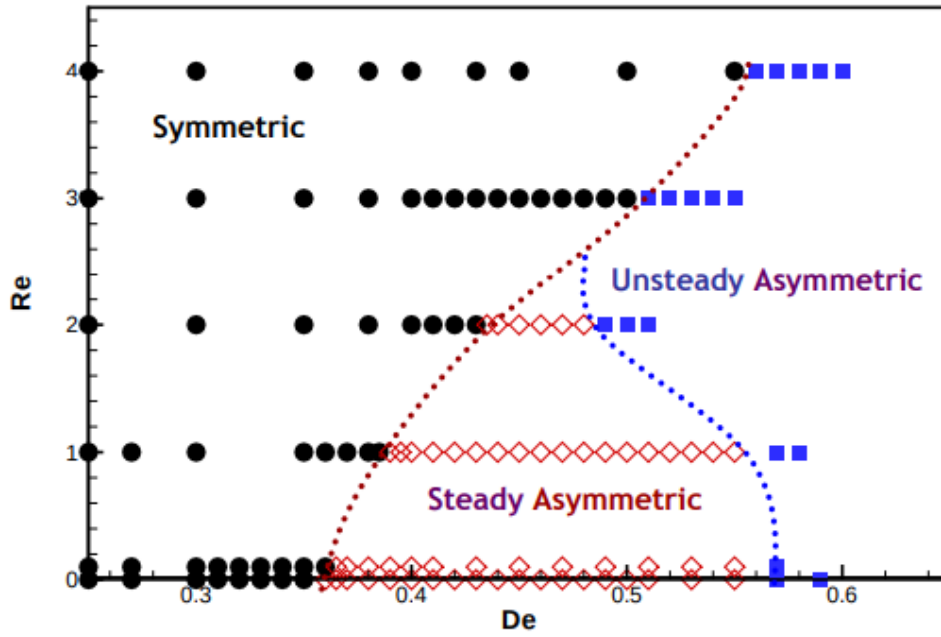


Figure 2.4: Re-De map for the Oldroyd-B viscoelastic fluid flow with $\beta = 1/9$ [18].

Several investigations were also performed for different cross-slot settings. Flow behaviour was evaluated for the case of three inlet channels and one outlet channel and steady asymmetric cases were also reported [15]. The stability of viscoelastic stagnation flows in a half cross-slot geometry, or T-shaped microchannel (see Figure 2.5) was also evaluated by Soulages et al. [25] in two different settings, with and without cavity. These two types of geometry allowed to have both a free and a fixed stagnation point, and the three addressed flow patterns were observed in the first case, whereas for the fixed stagnation point, critical conditions were reported to be substantially lower with a direct transition from the steady symmetric flow to the three-dimensional time-dependent flow at a critical Weissenberg number.

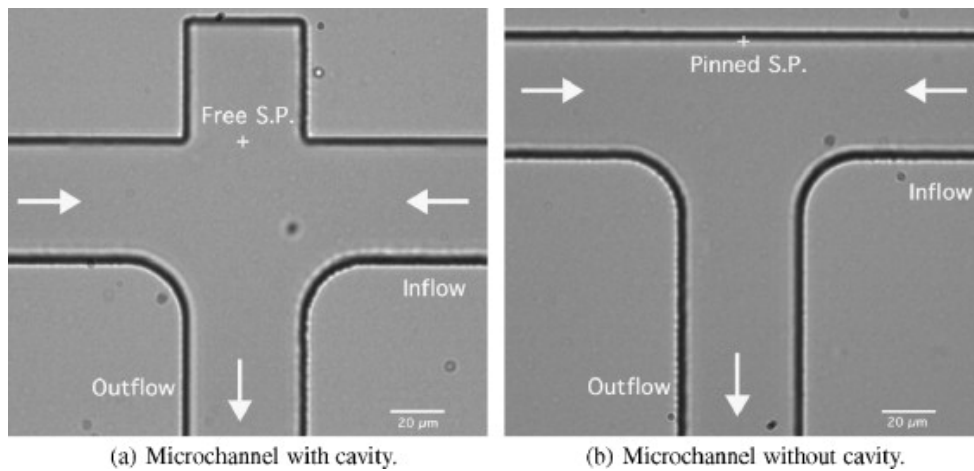


Figure 2.5: Optical micrographs of the microchannel (a) with and (b) without cavity. The anticipated location of the stagnation point (S.P.) sitting on a free streamline (a) or pinned onto the confining wall (b) is indicated in the channel. (a) Microchannel with cavity. (b) Microchannel without cavity [25].

2.5 Multiphase flows

The simultaneous presence of different phases in external and internal flows consisting of combinations of gas, liquid and solid is consistently found in daily life environments being termed multiphase flows.

The numerical simulations of multiphase flows date back to the start of computational fluid mechanics and are a problem of high importance and wide industrial application. These flows require specific numerical methods, one example being the work done by Sommerfeld [24], where an overview of numerical methods for the calculation of dispersed multiphase flows was presented. Multiphase flows including both viscoelastic and Newtonian fluids are also well documented in the literature. A typical problem is the viscoelastic droplet in a Newtonian medium (VN system) where the droplet interface assumes a dumbbell shape, or the Newtonian droplet in a viscoelastic medium where the droplet has an almost elliptical shape [8], [4] (see Fig 2.6). This study can also be recreated by following the impacting drop rheoTool tutorial [17]. The current methods used to simulate flows with a material interface can be divided in two different groups: surface and volume methods. In the surface methods the interface is explicitly tracked by aligning it with the computational mesh, programmed to advance with the interface and volume methods, having the advantage of knowing the interface position accurately and the disadvantage of being inefficient in handling large topological and interface movements [21]. In the volume methods an indicator function is used to identify the fluid on both sides of the interface, the major disadvantage this method introduces is the absence of the explicit determination of the interface location [21]. In any viscoelastic fluid phases combination, the accurate description of the flow dynamics requires

the combination of interface methods with the numerical techniques to deal with this type of fluids, such as the log-conformation method described in Chapter 3 [1].

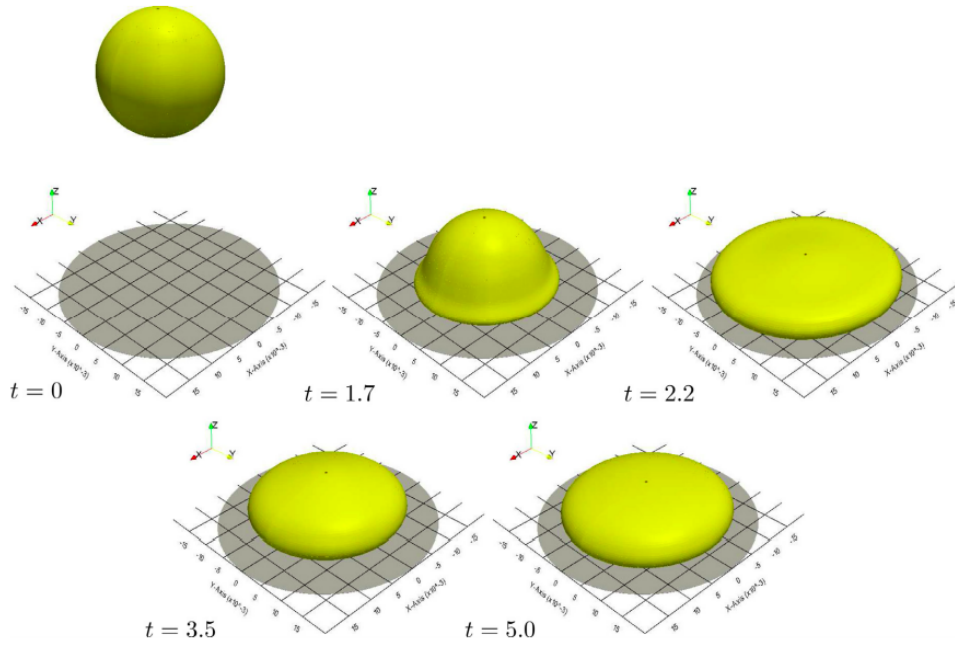


Figure 2.6: 3D view of droplet spreading time evolution by Figueiredo et al. [8]

Chapter 3

Fundamental Equations and Numerical Method

In this chapter the equations governing the fluid flow are introduced. It is also presented the constitutive equations for the constitutive models evaluated in this dissertation: the Upper-Convected Maxwell model (UCM) and the Oldroyd-B model; as well as their numerical implementations.

3.1 Introduction

For the characterization of a flow, the knowledge of the velocity ($\mathbf{u}$), pressure (p), temperature (T) and the stress state ($\boldsymbol{\tau}$) is required in every instant and position. The velocity is composed by 3 Cartesian components (u, v, w).

Regarding the stress, it can be characterized by a tensor, as long as its 6 independent components are known ($\tau_{xx}, \tau_{xy}, \tau_{yy}, \tau_{xz}, \tau_{zz}, \tau_{yz}$). This stress tensor can be written in a Cartesian coordinate system in the following way

$$\begin{bmatrix} \tau_{xx} & \tau_{yx} & \tau_{zx} \\ \tau_{xy} & \tau_{yy} & \tau_{zy} \\ \tau_{xz} & \tau_{yz} & \tau_{zz} \end{bmatrix} \quad (3.1)$$

Each column represents the x , y and z components of the stresses acting upon the face of a cubic material portion of infinitesimal volume, normal to the direction of the coordinate system axes, Figure 3.1. The first index identifies the actuation face by its normal, while the second index represents the stress direction. Out of the nine components of this tensor, only six are independent, being identical the stresses with permuted indexes due to the momentum equilibrium. The tensor is therefore symmetric.

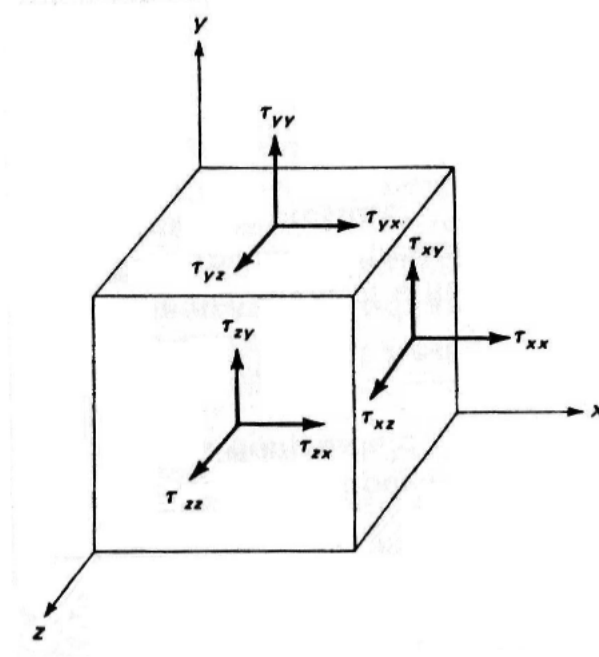


Figure 3.1: Graphical representation of the stress tensor in a Cartesian referential.

Up until now, to characterize a fluid flow in three dimensions, eleven variables would be necessary ($u_x, u_y, u_z, \tau_{xx}, \tau_{xy}, \tau_{yy}, \tau_{xz}, \tau_{zy}, \tau_{zz}, p$ and T). In a bi-dimensional case, the number of variables is reduced to seven ($u_x, u_y, \tau_{xx}, \tau_{xy}, \tau_{yy}, p$ and T). In this work, the problem is bi-dimensional and isothermal, being only six variables necessary.

3.2 Governing equations

The governing equations for this problem are conservation of mass, assuming incompressibility, and momentum:

$$\nabla \cdot \mathbf{u} = 0 \quad (3.2)$$

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \nabla \cdot \boldsymbol{\tau}' + \mathbf{f} \quad (3.3)$$

where $\mathbf{u}$ represents the velocity vector, t is time, p is the pressure, $\boldsymbol{\tau}'$ is the extra-stress tensor and $\mathbf{f}$ represents any exterior force.

The total number of equations, considering the conservation of mass and conservation of momentum is three, which are not enough for the solution of a fluid flow. It is now necessary to relate the stress and velocity fields of the flow, by recurring to a constitutive model.

3.3 Constitutive equations

For Newtonian fluids, the Newton's viscosity law,

$$\boldsymbol{\tau} = 2\eta_s \mathbf{D} = \eta_s [\nabla \mathbf{u} + (\nabla \mathbf{u})^T] \quad (3.4)$$

is adequate to relate the stress and velocity fields. This equation is composed by the viscosity of the fluid (η_s) and by the rate of deformation tensor ($\mathbf{D}$). Substituting the stress by Equation (3.4) in the conservation of momentum Equation (3.3), simplifies in the Navier-Stokes equation, that can be written in the scalar form along the i direction.

$$\rho \left(\frac{\partial u_i}{\partial t} + u_k \frac{\partial u_i}{\partial x_k} \right) = -\frac{\partial p_i}{\partial x_i} + \eta_s \frac{\partial}{\partial x_k} \left(\frac{\partial u_i}{\partial x_k} \right) \quad (3.5)$$

Therefore, the characterization of a laminar, incompressible and Newtonian fluid flow, requires the solution of a system of three equations, composed by the conservation of mass Equation (3.2) and by the Navier-Stokes Equation (3.5), where the variables are the pressure and the two velocity components. In order to know the stress field of the flow, one must substitute the velocity field in the constitutive Equation (3.4).

When dealing with viscoelastic fluid flows its usual to split the stress tensor in two terms relative to the solvent (τ_s) and relative to the polymer (τ), being that $\tau' = \tau + \tau_s$. It becomes now necessary to attribute constitutive equations to each tensor, defined by Equations (3.6) and (3.7), keeping the equation system closed:

$$\tau_s = \eta_s (\dot{\gamma}) [\nabla \mathbf{u} + (\nabla \mathbf{u})^T] \quad (3.6)$$

$$f(\tau) \tau + \lambda (\dot{\gamma}) \overset{\nabla}{\tau} = \eta_p (\dot{\gamma}) [\nabla \mathbf{u} + (\nabla \mathbf{u})^T] \quad (3.7)$$

where η_s represents the solvent viscosity, η_p represents the polymeric viscosity coefficient, λ represents the relaxation time and $\dot{\gamma}$ represents the deformation rate. $f(\tau)$ is a general scalar function that depends of an invariant of τ and $\overset{\nabla}{\tau}$ represents the upper-convected derivative, defined in the next section.

3.4 Constitutive models for viscoelastic fluid flows

In this section the Upper-Convected model (UCM) and the Oldroyd-B model are characterized. These models seek to mathematically describe viscoelastic fluid flow behaviour.

3.4.1 Upper-convected Maxwell model

This model proposed by Maxwell [13], corresponds to the first idealization of a viscoelastic fluid. Based on the behaviour of these fluids as simultaneously viscous and elastic, Maxwell empirically

defines his constitutive law through an analogy with a spring/damper mechanical system, Figure 3.2.

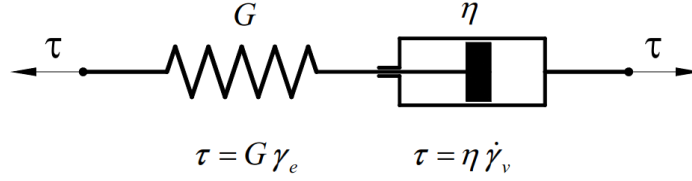


Figure 3.2: Mechanical model equivalent to a Maxwell fluid.

This analogy is based on the following: when velocities of low deformation are applied to the mechanical system, its response is purely dissipative, revealing the viscous response of the fluid; when a sudden solicitation is applied, the response of the system is purely elastic since the damper has no time to deform, revealing the elastic response of the fluid.

Since both elements are under the same stress, τ , the total deformation rate results in the sum of the deformation rates of the two elements, $\dot{\gamma} = \dot{\gamma}_e + \dot{\gamma}_v$, therefore having

$$\dot{\gamma} \eta_p = \frac{1}{G} \tau + \frac{1}{\eta_p} \tau \quad (3.8)$$

This expression can be written in the following manner

$$\dot{\gamma} \eta_p = \lambda \frac{\partial \tau}{\partial t} + \tau \quad (3.9)$$

where $\lambda = \eta_p / G$ represents the Maxwell's relaxation time. This equation is not acceptable as it is presented, because it violates the principle of material objectivity [14]. One way for this equation to verify the conditions proposed by Oldroyd [14] is to substitute the partial derivative $\partial \tau / \partial t$ with the Maxwell's upper convective derivative, $\overset{\nabla}{\tau}$, defined by [2]

$$\overset{\nabla}{\tau} = \frac{\partial \tau}{\partial t} + \nabla \cdot \mathbf{u} \tau - \tau \cdot \nabla \mathbf{u} - \nabla \mathbf{u}^T \cdot \tau \quad (3.10)$$

Resulting in the following constitutive equation:

$$\tau + \lambda \left(\frac{\partial \tau}{\partial t} + \nabla \cdot \mathbf{u} \tau \right) = \eta_p (\nabla \mathbf{u} + \nabla \mathbf{u}^T) + \lambda (\tau \cdot \nabla \mathbf{u} + \nabla \mathbf{u}^T \cdot \tau) \quad (3.11)$$

which is a quasi-linear viscoelastic model, used frequently in viscoelastic fluid flows simulations. This model is the Upper-Convected Maxwell model (UCM) and predicts constant values for the shear viscosity and the first normal stress difference, and a null value for the coefficient of the second normal stress difference. Despite the attractive simplicity of this model, it is not the most suitable for the modeling of most viscoelastic fluids. It does not reproduce the effect of the second normal stress difference and does not reflect the correct physical behaviour in certain

conditions of elongational flow. Numerically, this model is difficult to handle, being often used in computational rheology test the robustness of new numerical methods.

3.4.2 Oldroyd-B model

The equation of the Oldroyd-B or Jeffreys model [11] is obtained from the Maxwell model through the inclusion of the temporal derivative of the deformation rate, maintaining a linear relation

$$\lambda \frac{\partial \tau}{\partial t} + \tau = \eta_0 \left(\dot{\gamma} + \lambda_r \frac{\partial \dot{\gamma}}{\partial t} \right) \quad (3.12)$$

where λ_r represents the retardation time. This equation can also be obtained by representing an analogous mechanical system, as shown in the following figure.

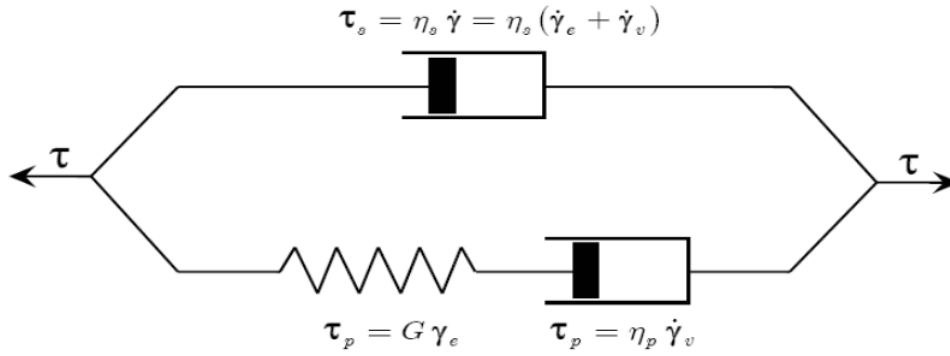


Figure 3.3: Mechanical model equivalent to the Oldroyd-B model.

Using the same methodology as used in the Maxwell's model, the total stress for the represented mechanical system is the sum of the stresses of each branch.

$$\tau = \tau_p + \tau_s \quad (3.13)$$

Note that the stress τ_p is defined through the equation of the Maxwell's model, and adding the stress for the damper in the superior branch, given by:

$$\tau_s = \eta_s \dot{\gamma} = \eta_s (\dot{\gamma}_s + \dot{\gamma}_v) \quad (3.14)$$

being $\eta_0 = \eta_p + \eta_s$ and $\lambda_r = \lambda \eta_s / \eta_0$. This model does not verify the principle of material objectivity as well. This requires a generalization for a generic coordinate system, like it was made for the UCM model. The Oldroyd-B model is now defined by the following equation:

$$\tau + \lambda \overset{\nabla}{\tau} = \eta_0 (\dot{\gamma} + \lambda_r \overset{\nabla}{\dot{\gamma}}) \quad (3.15)$$

Where the upper convective derivatives for the stress, $\overset{\nabla}{\tau}$, and for the deformation rate, $\overset{\nabla}{\dot{\gamma}}$, are calculated by Equation (3.10) of the convective derivatives presented in the previous sub-chapter.

This model can also be described in another way, that consists of considering the total stress as the sum of the contributions of polymeric nature and the contributions of a purely viscous nature, that results in the following equations:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_s + \boldsymbol{\tau}_p \quad (3.16)$$

$$\boldsymbol{\tau}_s = \eta_s (\nabla \mathbf{u} + \nabla \mathbf{u}^T) \quad (3.17)$$

$$\boldsymbol{\tau}_p + \lambda \left(\frac{\partial \boldsymbol{\tau}_p}{\partial t} + \nabla \cdot \mathbf{u} \boldsymbol{\tau}_p \right) = \eta_p (\nabla \mathbf{u} + \nabla \mathbf{u}^T) + \lambda (\boldsymbol{\tau}_p \cdot \nabla \mathbf{u} + \nabla \mathbf{u} + \nabla \mathbf{u}^T \cdot \boldsymbol{\tau}_p) \quad (3.18)$$

The Oldroyd-B model predicts constant values for the shear viscosity and for the first normal stress difference, and a null value for the second normal stress difference.

The equations presented define the system of equations to be solved. However, due to problems regarding numerical stability, the system cannot be solved with these equations alone, and several numerical stabilization techniques are presented ahead.

3.5 Numerical method

The open source software OpenFOAM, used in this study, is one of the most popular softwares in computational fluid mechanics and is based in the Finite Volumes Method. With this framework, the computational domain is split in several control volumes where the variables of the flow are calculated. The number of computational cells define the mesh refinement and validity of results. For an orthogonal mesh, the cell where the governing equations are calculated is designated by cell P, and its neighbour cells are designated through a geographical notation that can be observed in Figure 3.4, or through the index F, in a more generic notation.

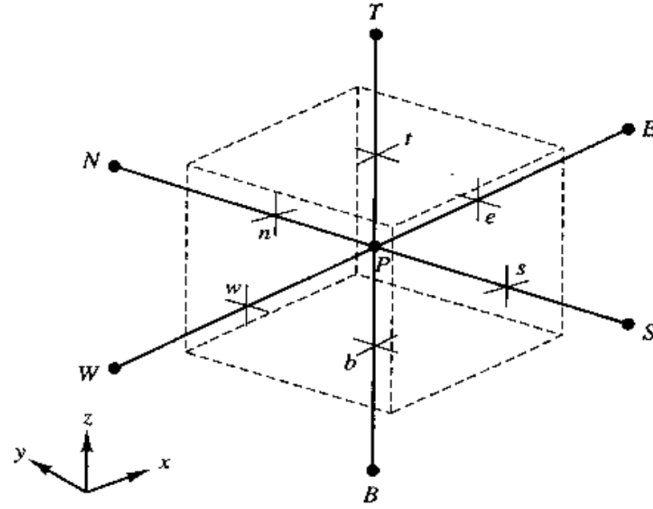


Figure 3.4: Control volume and faces orientation. W (west), E (east), S (south), N (north), B (bottom), T (top).

The meshes used in this work are collocated meshes, meaning that every variable is calculated in the same representative point of the cell, which is normally located at its geometrical center (node P and neighbours F). This implies the need for a special interpolation in the calculation of the flow rate that travels through the faces of the control volume. The interpolation schemes used are mentioned more ahead.

3.6 Numerical stabilization for viscoelastic fluid flows simulations

As aforementioned, the governing equations solved in this numerical study are not enough to guarantee numerical stability, as it is necessary to apply specific techniques and interpolation schemes in order to solve viscoelastic fluid flows with higher Weissenberg numbers, one of the main focuses of this dissertation. In the OpenFOAM software the *viscoelasticFluidFoam* solver developed by Favero et al. [9] aims to model viscoelastic fluids, the problem with this solver arises at high Weissenberg numbers, where it becomes unsteady. Pimenta et al. [16] [17] minimize these issues, and the modifications performed in the *viscoelasticFluidFoam* solver are implemented and available in the rheoTool package, and presented in this sub-chapter.

3.6.1 The log-conformation tensor approach

One of the main difficulties in viscoelastic fluid flows simulations, is the numerical instability existent in high Weissenberg number flows. Being these flows an important part of this study, it becomes essential to expose the approach of the log-conformation technique [16] [17], elaborated with the goal of reducing these instabilities.

The log-conformation method consists in a change of variable when evolving the polymeric extra-stress in time, relating it with the conformation tensor ($\mathbf{A}$). In the constitutive models used in this dissertation, this relation is presented by the following equation,

$$\boldsymbol{\tau} = \frac{\eta_p}{\lambda} (\mathbf{A} - \mathbf{I}). \quad (3.19)$$

It is defined a new tensor ($\boldsymbol{\Theta}$) as the natural logarithm of the conformation tensor, that can be diagonalized, since its a positive matrix defined as

$$\boldsymbol{\Theta} = \ln(\mathbf{A}) = \mathbf{R} \ln(\boldsymbol{\Lambda}) \mathbf{R}^T \quad (3.20)$$

where $\mathbf{R}$ is a matrix containing the eigenvectors of $\mathbf{A}$ in its columns and $\boldsymbol{\Lambda}$ is a matrix whose diagonal elements are the respective eigenvalues resulting from the decomposition of $\mathbf{A}$. Rewriting the constitutive Equation (3.7), in terms of $\boldsymbol{\Theta}$ it becomes,

$$\frac{\partial \boldsymbol{\Theta}}{\partial t} + \mathbf{u} \cdot \nabla \boldsymbol{\Theta} = \boldsymbol{\Omega} \boldsymbol{\Theta} - \boldsymbol{\Theta} \boldsymbol{\Omega} + 2\mathbf{B} + \frac{1}{\lambda} g(\boldsymbol{\Theta}) \quad (3.21)$$

in which $g(\boldsymbol{\Theta})$ is a model-specific tensorial function depending on $\boldsymbol{\Theta}$ and

$$\mathbf{B} = \mathbf{R} \begin{bmatrix} m_{xx} & 0 & 0 \\ 0 & m_{yy} & 0 \\ 0 & 0 & m_{zz} \end{bmatrix} \mathbf{R}^T \quad (3.22)$$

$$\boldsymbol{\Omega} = \mathbf{R} \begin{bmatrix} 0 & \omega_{xy} & \omega_{xz} \\ -\omega_{xy} & 0 & \omega_{yz} \\ -\omega_{xz} & -\omega_{yz} & 0 \end{bmatrix} \mathbf{R}^T \quad (3.23)$$

$$\mathbf{M} = \mathbf{R} \nabla \mathbf{u}^T \mathbf{R}^T = \begin{bmatrix} m_{xx} & m_{xy} & m_{xz} \\ m_{yx} & m_{yy} & m_{yz} \\ m_{zx} & m_{zy} & m_{zz} \end{bmatrix} \mathbf{R}^T \quad (3.24)$$

$$\omega_{ij} = \frac{\Lambda_j m_{ij} + \Lambda_i m_{ji}}{\Lambda_j - \Lambda_i} \quad (3.25)$$

After solving Equation (3.21), $\boldsymbol{\Theta}$ can be diagonalized in the following manner

$$\boldsymbol{\Theta} = \mathbf{R} \boldsymbol{\Lambda}^\Theta \mathbf{R}^T \quad (3.26)$$

and by the inverse relation of Equation (3.20) it is possible to recover the conformation tensor

$$\mathbf{A} = \exp(\boldsymbol{\Theta}) = \mathbf{R} \exp(\boldsymbol{\Lambda}^\Theta) \mathbf{R}^T \quad (3.27)$$

Finally the polymeric extra-stress tensor can be computed from $\mathbf{A}$, Equation (3.19), and used in the momentum equation.

3.6.2 Pressure-velocity coupling algorithm

Applying the continuity equation to the pressure variable and following the SIMPLEC methodology [16] [17] leads to,

$$\nabla \cdot \left(\frac{1}{a_p - H_1} (\nabla p)_p \right) - \nabla \cdot \left[\frac{\mathbf{H}}{a_p} + \left(\frac{1}{a_p - H_1} - \frac{1}{a_p} \right) (\nabla p^*)_p \right] \quad (3.28)$$

where a_p represents the diagonal coefficients from the momentum equation, H_1 represents the symmetrical value of the sum of the off-diagonal coefficients from the momentum Equation (3.29), $\mathbf{H}$ represents the off-diagonal contributions (3.30) plus source terms and p^* is the pressure field calculated in the iteration, or previous time step.

$$H_1 = - \sum_{nb} a_{nb} \quad (3.29)$$

$$\mathbf{H} = - \sum_{nb} a_{nb} \mathbf{u}_{nb}^* + \mathbf{b} \quad (3.30)$$

Once obtained the pressure field congruent with the continuity Equation (3.28) the velocity is corrected through the following equation.

$$\mathbf{u} = \frac{\mathbf{H}}{a_p} + \left(\frac{1}{a_p - H_1} - \frac{1}{a_p} \right) (\nabla p^*)_p - \frac{1}{a_p - H_1} (\nabla p)_p \quad (3.31)$$

3.6.3 Stress-velocity coupling

In Pimenta and Alves [16] [17], the Equation (3.32) was formulated for the divergent description of polymeric extra-stresses ($\nabla \cdot \boldsymbol{\tau}$), proving to be quite stable.

$$\boldsymbol{\tau}_f = \overline{\boldsymbol{\tau}}_f + \eta_p \left[(\nabla \mathbf{u})_f + (\nabla \mathbf{u})^T |_f - \left(\overline{\nabla \mathbf{u}} |_f + \overline{(\nabla \mathbf{u})^T} |_f \right) \right] \quad (3.32)$$

Where the terms with an over-bar are obtained through linear interpolation from cell-centered values, while the other velocity gradients are directly evaluated from the cell-centered velocities straddling the face. Introducing $\boldsymbol{\tau}_f$ as defined in the previous formulation, in the momentum equation, with both-sides-diffusion terms already present, results in:

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) - \nabla \cdot (\eta_s + \eta_p) \nabla \mathbf{u} = -\nabla p - \overline{\nabla \cdot \eta_p \nabla \mathbf{u}} + \nabla \cdot \overline{\boldsymbol{\tau}} + \mathbf{f} \quad (3.33)$$

Here, the term $\overline{\nabla \cdot \eta_p \nabla \mathbf{u}}$ is a "special second-order derivative" and represents the divergence of the velocity gradient, where the velocity gradient at the faces is obtained by linear interpolation of the velocity gradient calculated at the cell centers. In the literature [16] it is also demonstrated that by refining the mesh, Equation (3.32) tends to $\boldsymbol{\tau}_f = \overline{\boldsymbol{\tau}}_f$, and the added terms cancel each other out. Note that in Equation (3.33) the transposed velocity gradients were ignored for simplicity, since the continuity determines that $\nabla \cdot \nabla \mathbf{u}^T = 0$.

3.6.4 High-resolution schemes

The discretization of convective terms in the finite-volume framework, leads to

$$\int_V (\mathbf{u} \cdot \nabla \phi) dV = \sum_f \phi_f (\mathbf{u}_f \cdot \mathbf{S}_f) = \sum_f \phi_f F_f \quad (3.34)$$

where ϕ is a generic advected variable, $\mathbf{S}_f$ is the face-area vector and F_f is the volumetric flux crossing the f face. The fluxes are already known through the interpolation presented in Equation (3.32), but in order to calculate ϕ in the face centers, another interpolation needs to occur using the already known cell-center values. In the OpenFOAM software, countless possible schemes are available to solve this calculation, but the high-resolution schemes are the ones chosen by Pimenta and Alves [16] [17] to be used with rheoTool and, consequently, in this study. In the HRS methods, the normalized weighting factor is the utilized approach, using dimensionless quantities defined as

$$\tilde{\phi}_f = \alpha \tilde{\phi}_C + \beta \quad (3.35)$$

and with the following relations

$$\tilde{\phi}_f = \frac{\phi_f - \phi_U}{\phi_D - \phi_U} \quad (3.36)$$

$$\tilde{\phi}_f = \frac{\phi_C - \phi_U}{\phi_D - \phi_U} \quad (3.37)$$

For better understanding of the indexes used in these equations, the norm is: for any face f , cell C represents the cell from where the flux comes (upstream), cell D (downstream) is the cell to which the flux goes and cell U (far-upstream) is the cell upstream to cell C. Usually, the cell U should not be identified as independent and ϕ_U can be defined as

$$\phi_U = \phi_D - 2(\nabla \phi)_C \cdot \mathbf{d}_{CD} \quad (3.38)$$

The α and β coefficients in Equation (3.35) are specific to each high-resolution scheme and can be functions of $\tilde{\phi}_C$. Since the scheme used in this work is the CUBISTA [16] [17], and the scalars take the following values:

$$[\alpha, \beta] = \begin{cases} [1, 0] & \tilde{\phi}_C \leq 0 \vee \tilde{\phi}_C \geq 1 \\ [7/4, 0] & 0 < \tilde{\phi}_C < 3/8 \\ [3/4, 3/8] & 3/8 \leq \tilde{\phi}_C \leq 3/4 \\ [1/4, 3/4] & 3/4 < \tilde{\phi}_C < 1 \end{cases} \quad (3.39)$$

The discretization of the upstream contribution (upwind) of the HRS is implicit while the discretization of the rest is explicit [16], which adding to the Equations (3.35 - 3.38) results in

$$\phi_f = [\phi_C]_{implicit} + [(\alpha - 1)\phi_C + \beta\phi_D + (1 - \alpha - \beta)(\phi_D - 2(\nabla\phi)_C \cdot \mathbf{d}_{CD})]_{explicit} \quad (3.40)$$

3.6.5 Log-conformation application to the UCM and Oldroyd-B models

If the function $f(\tau)$ in Equation (3.7) is unitary, that relation describes the UCM model, that can be used to represent the simplified behaviour of a molten polymer.

$$\tau + \lambda \overset{\nabla}{\tau} = \eta_p [\nabla \mathbf{u} + (\nabla \mathbf{u})^T] \quad (3.41)$$

After applying the log-conformation, the constitutive equation takes the following form

$$\frac{\partial \Theta}{\partial t} + \mathbf{u} \cdot \nabla \Theta = (\Omega \Theta - \Theta \Omega) + 2\mathbf{B} + \frac{1}{\lambda} (e^{-\Theta} - \mathbf{I}) \quad (3.42)$$

If the contribution of a Newtonian solvent, τ_s (3.6), is added to the tensor τ , the Oldroyd-B model is obtained. In this model a parameter β is commonly referred, and it represents the viscosity ratio of the fluid, defined as

$$\beta = \frac{\eta_s}{\eta_s + \eta_p} \quad (3.43)$$

Despite the limitations of these models, namely the incapacity to predict the viscosity variation in the flow and a molecular extensibility limit, they are capable of qualitatively capture some of the characteristics of elastic flows.

3.6.6 Multiphase specifications

The multiphase calculations performed by rheoTool use the volume of fluid method (VOF) that defines the phases occupying a region of space by a phase fraction. When two phases are separated a sharp phase fraction transition occurs at their interface from 0 to 1. This method requires a phase based interpolation in order to calculate the interface position, which was implemented in this work as a post processing tool that can be consulted in Chapter 4. In terms of constitutive equations of the flow, several variables come described by a function of each defined phase, for example, the density value attributed to a mesh point at a certain time, is characterized by the sum of every considered fluid density multiplying by their respective phase fraction as a weighting term. The momentum and other fluid properties are of the "mixture" and a single momentum equation is solved.

Chapter 4

Software Description

The numerical simulations in this study were performed using OpenFOAM, the most popular open-source software in computational fluid mechanics in the engineer and scientific fields. This tool is based on the finite volume method, and consists of a set of C++ libraries, with several solvers that solve the characteristic equations specific to each flow, complemented with some utilities that allow for example, the mesh creation and data post-processing.

The solver used in this study is the *rheoInterFoam*, using the *SIMPLEC* algorithm for pressure-velocity coupling, available in the *rheoTool* package developed by Pimenta and Alves [17]. This tool is used to study multiphase viscoelastic fluid flows, and is based in the *interFoam* solver, available in the OpenFoam software, to which several numerical stabilization techniques were applied, allowing for example, a high Weissenberg number flow simulation.

In this chapter the main characteristics of the OpenFOAM software are exposed, as well as the required steps to run a simulation for this study. The post-processing method, as well as the software developed to do so, are also showcased.

4.1 Folder organization

The data of every OpenFOAM study case is contained in a directory (Figure 4.1), through a series of organized files in three types of sub-directories:

- *constant*, that contains a sub-directory named *polyMesh* where the mesh is defined. Here, the dictionary *constitutiveProperties* is also found. This dictionary is exclusive to the *rheoTool* software, and here the fixed simulation parameters such as physical properties and information regarding the constitutive models are present. An example of this file is presented in Appendix A.

- *system*, where the files regarding simulation and numerical method control are found. The main dictionaries found in this directory are *controlDict*, that allows to control the time-step definitions, the simulation time and the Courant number limit (explained ahead). *fvSchemes* contains the differential equations discretization methods and *fvSolutions* is where the equation solvers are

defined. Here is also found the dictionary *decomposeParDict* where one can specify the number of processors to use in the simulation and *setFields* where the specification of the starting location of each fluid phase is made.

- *time directories*, is where each calculated variable field is registered for the specified instants of time in *controlDict*. Before running the simulation there is only the time directory relative to the initial instance, nominated 0 , where the user defines the flows initial conditions. As referred in chapter 3, the rheoTool package makes use of the logarithmic conformation method, and for that reason a new field needs to be considered, representative of the conformation tensor natural logarithm (Θ), besides the typical fields existent in the original solver of the OpenFOAM software. These are p , U , τ and α , relative to the pressure, velocity, stress and phase respectively. The eigen vectors are also calculated, *eigVecs* (represented by $\mathbf{R}$) and the eigen values, *eigVals* (represented by Λ), that result from the diagonalization of the conformation tensor. Despite not being necessary to define these last two fields in the initial instance, they are read if present.

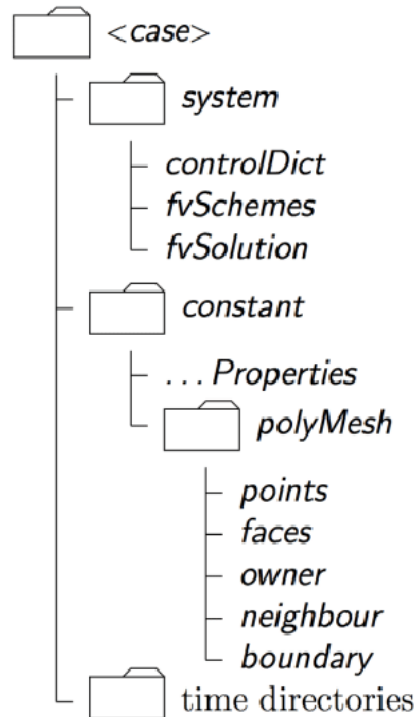


Figure 4.1: Directory structure of a general OpenFOAM case (figure from the manual [10]).

4.2 Pre-processing

4.2.1 Mesh criation

As previously established in chapter 3, the numerical simulations performed by OpenFOAM are based in the finite volumes method, as it is necessary to represent the physical domain in a properly defined mesh.

The geometrical definition of the mesh is found in the *blockMeshDict* file of the *polyMesh* sub-directory, where the user can create a mesh from scratch, or alter a previously created one. To create a mesh every geometry vertices are first defined through cartesian coordinates, followed by the block creation. The blocks are geometrical structures containing eight vertices that are divided in several cells according to a given description. With the increasing of cell numbers imposed the mesh becomes more refined, which translates into longer simulations in time and higher precision of results. It is also possible to add an expansion gradient to the cells in a given direction. This can be advantageous because it allows the user to have a more flexible mesh refinement, varying from a physical domain region to another. This process and its intricacies are exposed in Chapter 5. The edges of the blocks are by definition, straight lines, but can also be represented by curves.

4.2.2 Boundary conditions

In the *blockMeshDict* file its possible to divide the geometry boundaries in several regions or patches, to which a label is associated in order to define the type of boundary condition of that region with the fictitious exterior.

Using these labels, the boundary conditions and initial fields are possible to define for each patch, in the starting instant of time directory, closing the equations system that governs the flow. The principal types of boundary conditions used in this study are *fixedValue*, that imposes a fix value over a surface of the patch to all time-steps of the simulation (Dirichlet boundary condition) and *zeroGradient*, that imposes a null gradient to the field across the entire surface of the patch (Neumann boundary condition). [3]

4.2.3 Discretized equations and solution

In Chapter 3, is clarified that flow description follows a series of partial differential equations.

Once these equations depend on spatial and temporal dimensions of continuous character, to be solved in computational fluid dynamics its necessary to substitute them with discrete approximations. Through this process, the discretization, the variables become known for specific time instances and for a finite number of spatial positions, which translates into a certain error. The numerical method applied should reduce this error to acceptable amounts, even though this also depends on the mesh refinement in space and time, besides the numerical order of the discretization methods.

In the *fvSchemes* file of the *system* sub-directory the finite volumes discretization methods, used to solve the derivatives and center-face interpolations, are specified. In this sub-directory the file *fvSolutions* is also found, where the algebraic equation solvers are defined. The content of these files is also presented in Appendix A.

4.2.4 Simulation control

The last step to run a simulation in the OpenFOAM software is to define the execution parameters of the solver, namely, the initial and final time instances of the simulation, the time-step between calculations and the time interval in which the existent fields are written and its precision. All of these definitions can be set from the *controlDict* dictionary, in the *system* sub-directory. The attribute *coded functionObjects* can also be defined in this file, allowing to extract values regarding variables of interest, for each time-step, while the simulation is running.

To reach numerical stability and in order to the variable fields converge to acceptable values, its necessary to impose a parameter for each cell, the Courant number [5], that relates the time-step ∂t , the magnitude of the cell velocity $|\mathbf{u}|$, and the cell size in the direction of the velocity, ∂x , in the following way

$$Co = \frac{\partial t |\mathbf{u}|}{\partial x} \quad (4.1)$$

In order to have stability and good solution convergence, a maximum value of order 1 is imposed to the Courant number for the entire mesh. A common practice is to determine the maximum ∂t value for the worst case possible, meaning the smaller cell where the velocity is higher, guaranteeing $Co < 1$ for the whole domain. It is also possible to run simulations with an adjustable time-step, where a new ∂t is calculated at each iteration, having a maximum Courant number, defined by the user, into account.

4.3 Starting a simulation

Having finalized the pre-processing, the user can run a simulation inserting the name of the solver (*rheointerFoam*, in the present work) in the command window or terminal. An OpenFOAM simulation can run in two ways: in forefront, where all of the simulation progress is presented in the terminal window; or in the background, allowing to save the simulation information in a file, keeping the terminal window free to perform other tasks. Another specification set in this stage is the number of processors performing the simulation. This number needs to be previously set in the *decomposeParDict* file present in the *system sub-directory*. The number of selected processors needs to have into account two parameters: the number of processors the device performing the simulation predisposes; and the communication speed between processor cores. Increasing the number of processors normally represents a decrease in simulation time, this is explained by the added computational power but is not a general rule, as the communication time between processor cores can result in a total decrease of the calculation speed.

4.4 Post-processing

Included in the OpenFOAM installation, the graphical software ParaView disposes of several tools capable of interacting with the mesh in 3D and observe the fields of each variable in any time instance associated to a time directory saved during the simulation.

Despite the ParaView tool allied with the *controlDict* programming making possible every data manipulation, parallel programs were developed for this body of work in order to facilitate this analysis. In this section the main calculations implemented in these programs are presented and the code can be consulted in Appendix B.

4.4.1 Flow steadiness

For this dissertation, the knowledge of the flow state, namely the steadiness or unsteadiness of the flow over time, is an important distinction to be made in every simulated flow. The methods to characterize this behaviour are many. One can observe several parameters evolution over time with the ParaView software and infer the state of their evolution. This tool allows an even more effective way to track the evolution of a strategic parameter in a specified point of the flow and from that evolution conclude about the flow state. In this dissertation, the quantity U_x/U_{mag} is tracked, where U_x represents the x component of the velocity in the point located in the center of the geometry and U_{mag} being the average velocity of the flow. These quantities were written in a file using the aforementioned *controlDict* programming , and an example of this representation can be visualized in Figure 4.2

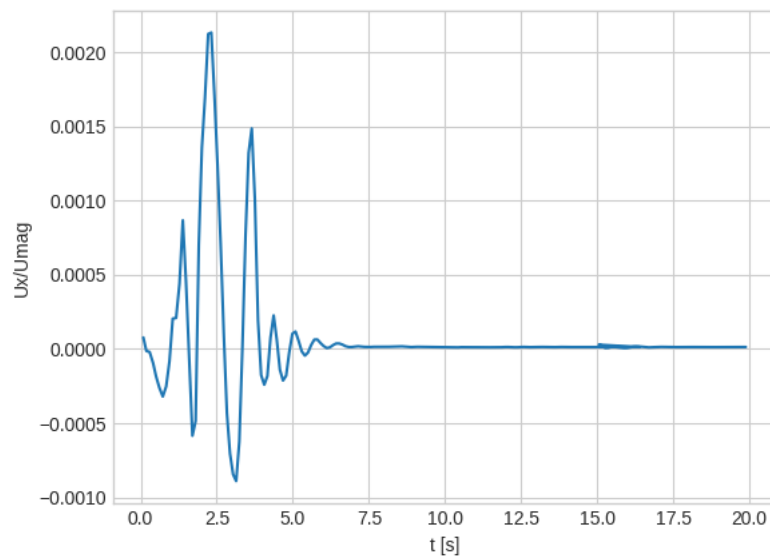


Figure 4.2: Example of the U_x/U_{mag} evolution over time.

In Figure 4.2 a good example of a steady flow is showcased. The velocity stabilization observed over time indicates that this flow is steady and could be analysed in the 10 seconds time-step for instance.

This tool is semi-independent as it takes advantage from the *controlDict* programming, the following showcased tools are the ones present in the Appendix B, as they are independent of the preexisting analysis programs and its calculations are always performed with the last time-step field files only.

4.4.2 Velocity profile

Another tool developed to better understand the underlying flow physics is the velocity profiling in specified areas. With the purpose of adding to this analysis, the velocity profiles were evaluated in the beginning of the cross-slot outlets and their module was used so that if the overlap of both curves coincides with one another the flow is symmetric and if they don't the flow is asymmetric.

4.4.3 Interface representation

Since the study at hand mainly consists of analysing a biphasic fluid flow of two immiscible fluids, their interface representation becomes important in the analysis process. With that in mind this tool was developed, also providing the knowledge of the stagnation point position. Since every point in the flow field has an associated variable regarding its phase (*alpha*) being a number from 0 to 1. Note that one fluid is defined by a $\alpha = 0$ and the other by $\alpha = 1$, having a value between 0 and 1 indicates that both fluids are present in the point's vicinity and that the interface must be in that region. Based on this concept the program was developed and an interpolation of the *alpha* values was performed between the focus points, regarding their position, in order to find the placement of the interpolated region where $\alpha = 0.5$. The stagnation point is found at the interface and corresponds to the point with zero velocity magnitude. An example of this representation can be visualized in Figure 4.3.

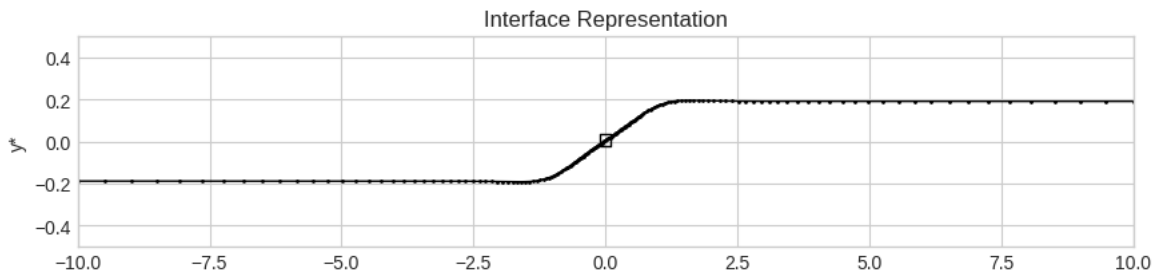


Figure 4.3: Interface representation for an asymmetric flow. The square represents the position of the stagnation point.

4.4.4 Local Weissenberg calculation

The local Weissenberg number is a simple but important calculation this thesis exposes very often. Since λ is a constant, the only particularity regarding its use is when we have two fluids with different relaxation times. In this case λ is set to be the linear interpolation between them, taking the stagnation point position and the closest northern and southern points from it, into account. Having the parameters set as in Figure 4.4, the calculation follows:

$$\lambda = \frac{\lambda_n(y_{stag} - y_s) + \lambda_s(y_n - y_{stag})}{y_n + y_s} \quad (4.2)$$

where y_{stag} represents the stagnation point y coordinate.

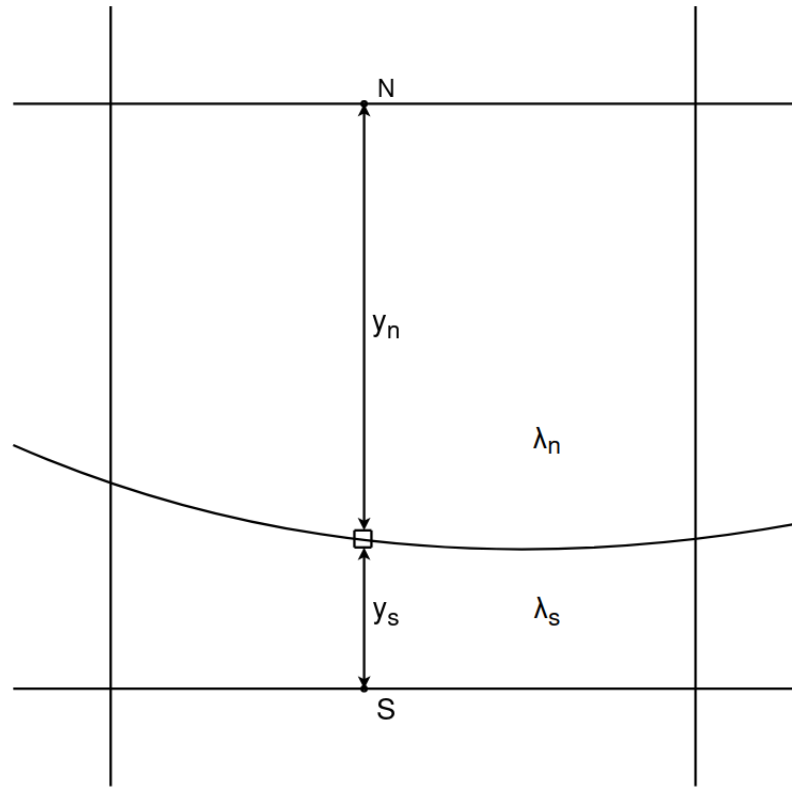


Figure 4.4: Variable illustration of the lambda calculation. The square represents the stagnation point location.

This way of obtaining the interpolated value of λ does require some additional calculations, mainly to know the N and S points location, however this method was preferred in detriment of a simple average calculation.

Finally, the calculation of the velocity components partial derivatives requires having the velocity and position parameters of the stagnation point neighbours and performing a simple discretization between them.

4.4.5 Stress plotting

The stress plotting in various point regions of the physical domain was also implemented. This graphical representation is not worth displaying as it is just a simple plotting of pre-written variables. The interesting characteristic is that one can manipulate these fields in order to calculate stress dependent quantities such as the normalized first normal stress difference for example.

$$N1^* = \frac{h(\tau_{yy} - \tau_{xx})}{2\eta_0 V} \quad (4.3)$$

Where h is a characteristic length, τ_{yy} and τ_{xx} are normal stresses in the y and x direction respectively, η_0 represents the zero-shear rate viscosity and V the average velocity at the channel inlet.

4.5 Final considerations

The complexity of equations and iterations involved in the present study case require the utilization of a computer with high processing capacity, and overall good performance, in order to complete the simulations in real or useful time.

To reduce this time, the access to the *High Performance Computing* (HPC) resources was made possible by FEUP. These consist of an organized set of clusters with similar technology to a normal computer but with more processors, more powerful CPU's, higher RAM and a network much faster than any consumer level one.

Another useful resource provided to the resolution of this body of work was a desktop computer access in CEFT, that turned out to be quite helpful not only as an additional simulation computer but also as a general tool available throughout the last few months.

Chapter 5

Mesh dependency study

In this section, the numerical accuracy of the flow calculations, using the generated block structured meshes is evaluated. This analysis represents one of the starting points to the simulations that lie ahead, since knowledge of the numerical uncertainty introduced, is of prime importance in order to establish a balance between computational resources and accuracy of results.

5.1 Mesh characteristics

For this analysis three levels of consistent mesh refinement are assessed, a base case domain and grid, which is obtained from the cross-slot tutorial in the rheoTool package [16], and its first and second refinements. The cross-slot two-dimensional (2D) geometry, is illustrated schematically in Figure 5.1.

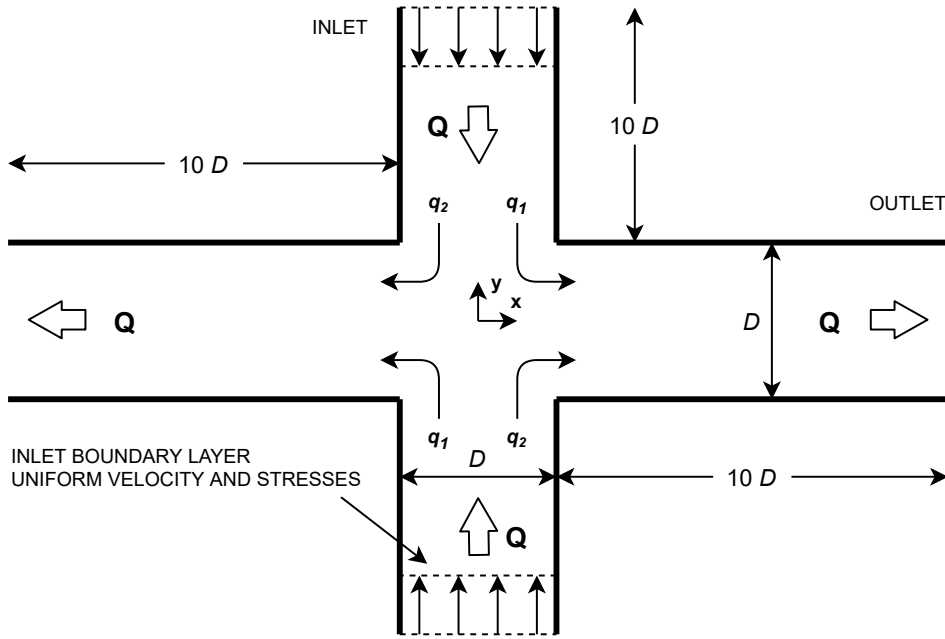


Figure 5.1: Schematics of the cross-slot geometry (Based on a figure found in [6]).

In terms of qualitative mesh properties two types of cell distribution are analysed, one having the same cell geometry along all inlets and outlets, and the other taking advantage of the Openfoam tool named *simpleGrading*. The terminology adopted in this thesis regarding both types of mesh, consists of identifying these with U and NU, for the uniform and non-uniform meshes respectively (see Table 5.1 for details). In Figure 5.2 the uniform mesh can be visualized.

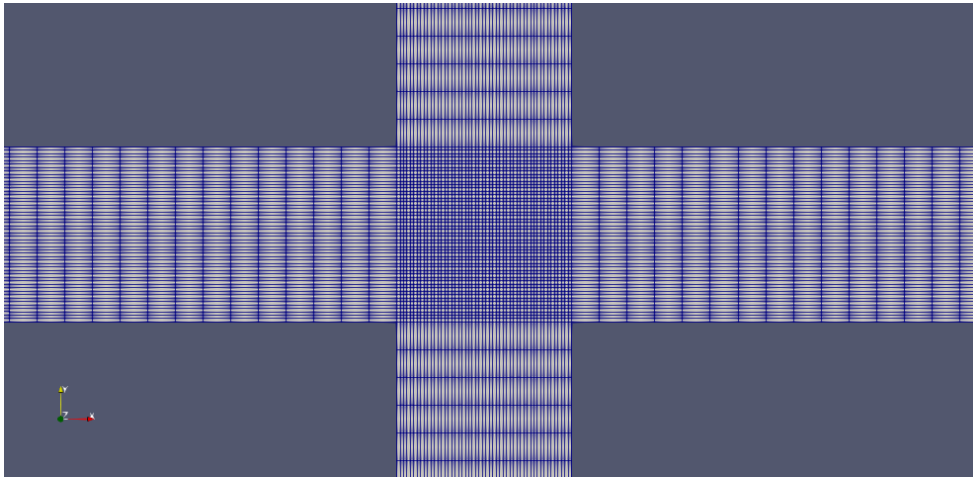


Figure 5.2: Schematics of the cross-slot unrefined geometry without cell size gradient.

Note that its cell size is smaller at the center, as it is the flow region with more intense variations in flow characteristics (see Table 5.1). For example, the highest velocity gradients are found here,

where the flow direction changes and therefore, the solution of the constitutive equations in this region should be privileged in order to level off the inaccuracies that these calculations present. In other words, the actual flow physics should be appropriately encapsulated. Simultaneously, in regions of slow variations of flow characteristics it is not necessary to have the same level of mesh fineness in order to have the same level of accuracy, i.e., coarser cells can be used where variations are weak. It is also a good practice to avoid abrupt changes between consecutive cells pertaining to different blocks, since this leads to a local reduction of accuracy that propagates to the whole domain.

In Figure 5.3, the cell size gradient mesh can be seen, as well as the respective blocks used to define it.

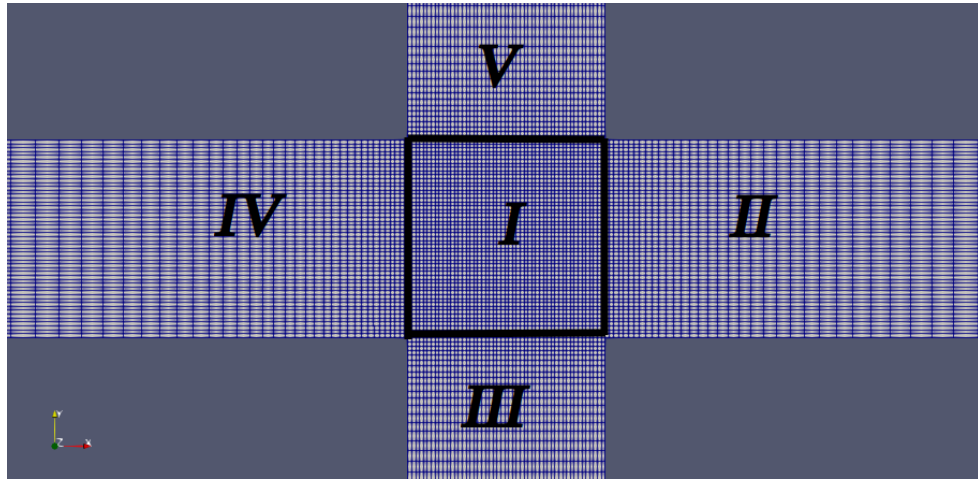


Figure 5.3: Schematics of the cross-slot unrefined geometry with cell size gradient and respective blocks used to generate the mesh.

In this case the cell length in the cross-slot arms gets increasingly longer in the streamwise directions, maintaining its geometry in the center block. This feature decreases the numerical error near the center square, the aforementioned focus area, in detriment of the outer cells calculation accuracy.

In terms of quantitative mesh characteristics, in each mesh refinement the number of computational cells doubles in each direction, except in the cross-stream direction where an additional cell ensures an odd number of control volumes and a set of cells along the symmetry plane. Even though the meshes are three dimensional, the assumption of two-dimensional flow means that grids only need to be refined in the flow plane, therefore at each mesh refinement level the number of control volumes is only increased by a factor of about four. In result, the accuracy of the simulation is also increased, as every FV method benefits from an increment of its cells / grid points.

Table 5.1: Characteristics of the computational meshes. NP - number of points; NC - number of cells; D - channel width

Mesh	NP	NC in Block I	NC in Blocks II/III/IV/V	$\frac{\Delta x_{min}}{D} = \frac{\Delta y_{min}}{D}$
M1-NU	30368	2601	3060	0.02
M2-NU	120098	10404	12240	0.01
M3-NU	477650	41616	48960	0.005
M1-U	30368	2601	3060	0.02
M2-U	120098	10404	12240	0.01
M3-U	477650	41616	48960	0.005

5.2 Mesh Analysis

For the mesh analysis, flow characteristics and specific study quantities need to be specified. The adopted approach consists of recreating previously done simulations. For this, the cross-slot analysis of Cruz et al. [6] provided numerous possibilities to base our mesh study on. As benchmarked in Cruz et al. [6], for the single-phase UCM model fluid with a selected Deborah number, $De=0.330$, guarantees steady flow development. As far as flow conditions go, the presence of a small Reynolds number, $Re=0.01$, represents the only difference this analysis has from its benchmark root where $Re \rightarrow 0$. This value was admitted into this study because, while maintaining the feasibility of the steady state flow assumption, it adds to the results obtained by Cruz et al., in variety and in possible physical interpretations. Note that a Reynolds number of 0.01 is extremely low, imposing little inertial effects on the flow.

The described simulations are analysed with some of the proposed benchmark variables [6], these being the local Weissenberg number Wi_0 and the flow asymmetry parameter DQ , proposed by Poole et al. [19]. The Richardson Extrapolation Method [22] is applied in order to increase the order of the truncation error, and further evaluate the numerical uncertainty corresponding to each mesh refinement. Note that in this case, a cell width gradient in the cross-slot arms is present, in order to be consistent with the benchmark study.

In Table 5.2. all results are presented for each mesh, as well as the extrapolated values.

Table 5.2: UCM model data for $De=0.330$. Wi_0 - local Weissenberg number; $|DQ|$ - flow asymmetry parameter; t - order of simulation time - t; ϵ_1 - local Weissenberg relative error; ϵ_2 - asymmetry parameter relative error;

Mesh	Wi_0	Wi_0 by Cruz et al.	$ DQ $	t (s)	ϵ_1	ϵ_2
M1-NU	0.523	0.509	0.250	10^4	0.2%	20,8%
M2-NU	0.533	0.502	0.295	10^5	2.1%	6,6%
M3-NU	0.526	0.500	0.310	10^6	0.8%	1.8%
Extrapolated value	0.522	0.499	0.312	—	—	—

The simulated results show good agreement in terms of local Weissenberg numbers. The general differences found relative to the results of Cruz et al. [6] could be explained due to the

different Reynolds numbers. The asymmetry parameter error is too high for it to be considered a reliable quantity to describe flow behaviour.

5.3 Mesh type comparison

The simulations performed in this thesis started from the simpler version of the mesh M1-U, but as it is good practice to avoid abrupt cell size differences the transition to the M1-NU mesh was made and this will be indicated ahead as results are presented.

In this subsection, the effects of cell width gradient in the cross-slot arms are evaluated in order to validate these initial simulations. If a significant difference is found between these mesh results, for the same simulation parameters, it is recommended the exclusive use of a cell size gradient. For the extensively analysed UCM/Newtonian analysis, present in the next chapter of this work, in a setting with a unitary Reynolds and zero surface tension, the critical Deborah and local Weissenberg numbers found are present in Table 5.3.

Table 5.3: UCM model/Newtonian data for $Re=1$ and in absence of surface tension.

	De	Wi_0
Mesh with gradient cell width	0.45	1.53
Mesh without gradient cell width	0.47	1.55

These results reflect a small difference in the critical Deborah number calculation, and quite negligible differences for the respective local Weissenberg number.

5.4 Conclusions

Measuring by numerical error introduced and in comparison to its benchmark root, the unrefined block mesh structure is considered apt for the local Weissenberg number calculation. However, the relative error of the asymmetry parameter $|DQ|$ cannot be overlooked and therefore, makes the use of this quantity as an analysis variable unreliable.

Even though the M2 and M3 meshes present better general results, the time resources this work predisposes favor the use of the coarser mesh, in detriment of one benchmark variable ($|DQ|$). Therefore the unrefined block mesh structure M1, will be the geometry used for all simulations in the present body of work.

In terms of mesh geometry, the uniform mesh revealed itself similar in results to its non-uniform counterpart. As mentioned before the uniform unrefined mesh geometry (M1-U) is utilized in the next chapter, as the slight differences found for the critical Deborah number can be overlooked since the obtained results aim to represent the physics at hand more than they do to serve bench-marking purposes, although in the inertialess crossing study, gradient cell width is present. Despite the aptitude of the uniform mesh, the gradient cell width mesh will be the one used in the rest of this dissertation since its results are theoretically more accurate.

Chapter 6

Interfacial flow between UCM and Newtonian fluids

6.1 Introduction

In this chapter several studies are performed for multiphase flows with Newtonian and viscoelastic fluids (described by the upper-convected Maxwell model). The main focus of this analysis is to quantify elastic instabilities for critical parameters, in order to understand their physical influence on the flow itself.

We will start by presenting an inertialess crossing, where the non-uniform block mesh (M1-NU) was used for benchmarking purposes, since this study is a starting point to the UCM/Newtonian flow. This benchmark is useful to this thesis, but to other works as well. This analysis is followed by the introduction of inertia effects, for the same Reynolds number in both inlets. Next we evaluate in more depth the effects of surface tension near the critical local Weissenberg values. This evaluation is mainly based on the capillary number (Ca) of the flow, as two critical Ca values appear for the $De=0.45$ case. Finally the study of several Reynolds number ratios between flow phases is the last analysis we present for this pair of fluids.

6.2 Inertialess flow

In this sub-chapter an evaluation of the aforementioned critical benchmark value [20] is presented, in order to create a base case for the UCM/Newtonian flow. We further complement this study presenting these results in both absence and presence of surface tension.

6.2.1 Simulation parameters

For this study both inertialess and low Reynolds number simulations were performed in a mesh with cell width gradient. In the latter a $Re=0.01$ was used whereas the achievement of a Stokes flow or creeping flow, is made by removing the advective term from the equation of conservation of momentum.

6.2.2 Results

The results obtained (see Table 6.1) show similar critical values when the capillary number varies from an unitary value to an infinite value (zero surface tension case). However, flow transition from a steady symmetric state to an unsteady state is significantly different and this may reflect a problem associated with dropping the advective term from the equation of conservation of momentum.

Table 6.1: Critical Deborah number (De) and local Weissenberg number (Wi_0) values.

		De	Wi_0
Re=0.01	Ca=1	0.40	1.21
	Ca= ∞	0.41	1.22
Re=0	Ca=1	0.44	1.32
	Ca= ∞	0.45	1.32

6.3 Unitary Reynolds flow

In this section the UCM/Newtonian flow is evaluated in the presence of inertial effects. The objective is to compare the results to the inertialess data and try to infer if increasing the Reynolds number to a unitary value, where inertial effects become significant, affects critical values or flow behaviour in a noteworthy manner. These simulations are performed in the previously described uniform mesh (M1-U).

6.3.1 Simulations performed

The simulations performed in this sub-chapter follow the previous ones, in terms of maintaining the same Reynolds number in both inlets. The main difference being that the actual Reynolds number is equal to one, thus introducing inertial effects. Also, the surface tension in these simulations is nonexistent ($Ca = \infty$). The relaxation time is the varying factor, determining the general Weissenberg number of the flow, or the Deborah number.

In the following table, the simulation range and respective flow states can be visualized. The obtained results indicate that the transition from a symmetric flow state happens directly towards an unsteady 3D flow.

Table 6.2: UCM model/Newtonian data for $Re = 1$ and $Ca = \infty$. * stands for unsteady flow.

Deborah Number	Local Weissenberg Number
0.10	0.32
0.31	0.98
0.33	1.05
0.34	1.08
0.35	1.11
0.38	1.20
0.47	1.55
0.48	*
0.49	*
0.50	*
0.51	*
0.55	*
0.60	*
0.70	*

These results, together with the inertialess cases, reflect the absence of a steady asymmetric flow transition, often observed in this thesis. In terms of critical values and in comparison to the inertialess cases, a small increase in the critical Deborah number is found but the local Weissenberg number of the respective flow is significantly higher. This suggests that inertial effects pose as a small contribution to either flow steadiness or unsteadiness but can actually result in significant Wi_0 variations indicating that the velocity gradients in the stagnation point region increase.

6.4 Capillary number effects

In this section the effects of the Capillary number are studied as it was considered to be an analysis of interest given the interface behavior. On one hand, the surface tension in the interface poses as additional normal stress in the stagnation point area, which can promote flow unsteadiness [19]. On the other, this tension has a stabilizing effect as it does, for example, in the cohesion of a water droplet.

To this end, near critical flows were taken into account to better understand the outcome of these diverging influences. The following chart contains all the Deborah and Capillary value pairs, with the exception of the infinite Capillary numbers, and their respective state (steady or unsteady), in order to create a map that reflects this duality of effects.

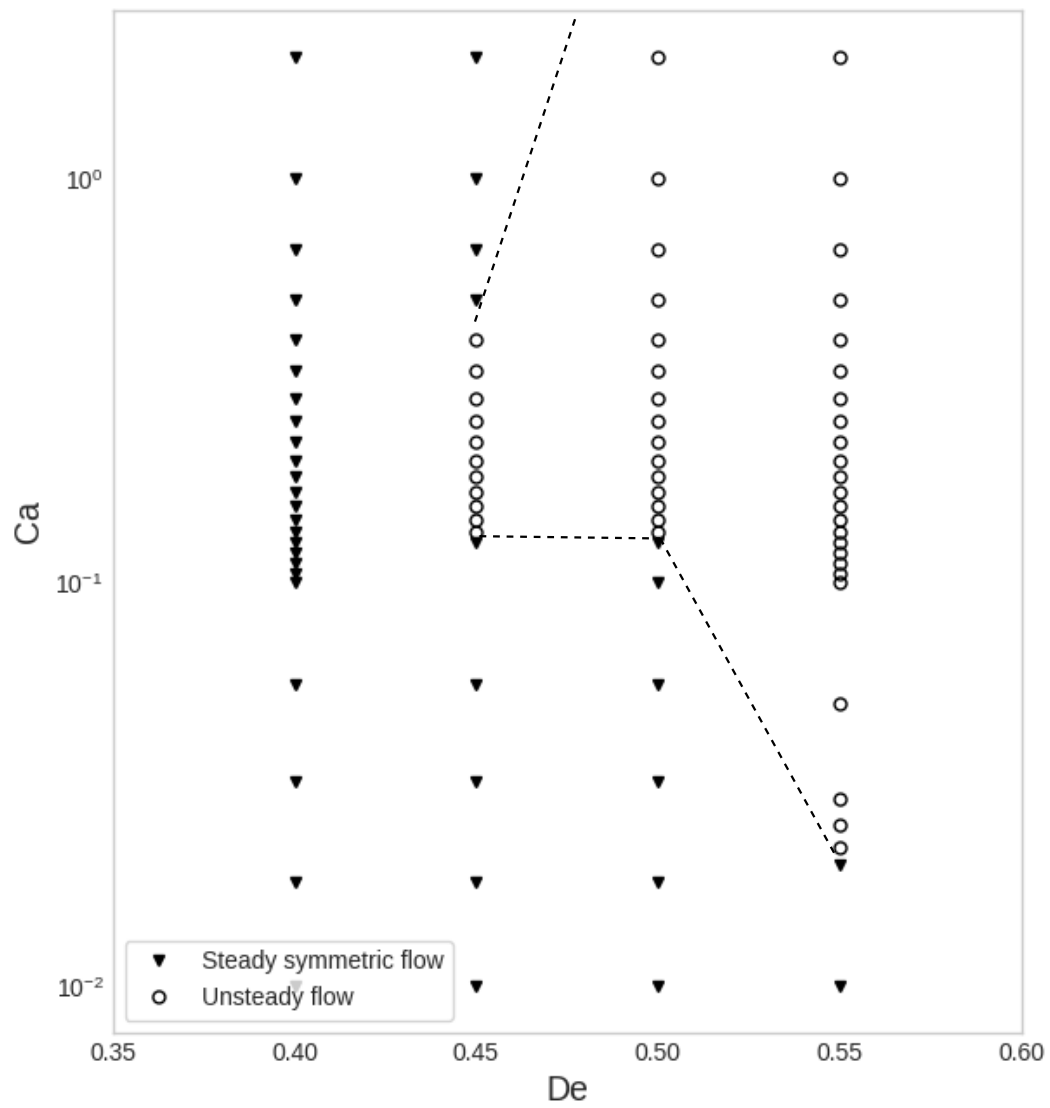


Figure 6.1: Ca - De phase diagram for UCM/Newtonian flows at $Re=1$.

Table 6.3: Local Weissenberg numbers for every simulated De and Ca pair. * represents unsteady flows, - represents not simulated pairs.

Wi_0		De			
		0.40	0.45	0.50	0.55
Ca	0.010	0.83	0.80	0.83	0.87
	0.018	1.29	1.51	1.32	-
	0.020	-	-	-	0.91
	0.022	-	-	-	*
	0.025	-	-	-	*
	0.022	-	-	-	*
	0.029	-	-	-	*
	0.032	1.20	1.43	0.83	-
	0.056	0.79	0.83	1.10	-
	0.100	0.76	-	0.76	*
	0.105	0.77	-	-	*
	0.111	0.79	-	-	*
	0.118	0.83	-	-	*
	0.125	0.92	1.03	0.79	*
	0.133	1.08	*	*	*
	0.143	1.16	*	*	*
	0.154	1.17	*	*	*
	0.167	1.20	*	*	*
	0.182	1.23	*	*	*
	0.200	1.25	*	*	*
	0.222	1.26	*	*	*
	0.250	1.27	*	*	*
	0.286	1.26	*	*	*
	0.333	1.26	*	*	*
	0.400	1.25	*	*	*
	0.500	1.24	1.44	*	*
	0.667	1.23	1.42	*	*
	1.000	1.22	1.41	*	*
	2.000	1.21	1.39	*	*
	20.00	1.19	1.38	*	*
	∞	1.30	1.53	*	*

The obtained results for the Deborah number of 0.45, show two critical Ca values of 0.125 and 0.500, in between which, the flow is unsteady. This reflects the duality of effects previously mentioned. For Ca numbers lower than 0.125, the surface tension between the two fluids could be considerable enough to stabilize their interface. This also occurs for De=0.50 and De=0.55, opening room for the investigation of this limit, that is suggested but not attended in this body of work.

Above the second critical value of Ca=0.500, the flow is also steady which is believed to be explained by the low or non existing increment on the normal stress difference in the stagnation point region [19]. For De=0.45, the surface tension involves a stress addition that can represent

the transition from a steady flow to an unsteady 3D flow. This effect is evidently irrelevant for a case like $De=0.55$ where a critical normal stress difference, should be already reached. And also for lower Deborah numbers like $De=0.40$, where this duality of effects seems to meet, since all simulated flows with $De=0.40$ are steady.

In order to investigate these stress related assumptions a normal stress total sum was made along the $x=0$ axis for the $De=0.45$ steady cases. The results found that for all flows with Ca values equal or lower than 0.125 the total normal stress sum was -43[Pa] and that for all flows with Ca values equal or higher than 0.500 this sum was -42[Pa]. No other tangible results could be found in other stress studies, being this stress variation between critical Ca values the only found correlation that further complements the supposition that this Ca duality of effects is stress related but not effectively explaining how.

The values that finish this mapping can only be obtained with more accurate simulations, which is suggested for future investigation.

6.5 Reynolds ratio consideration

This section intends to establish the relationship between the Reynolds ratio of the two fluids at hand, and their critical values. This ratio is achieved by considering different densities for both fluids and it is calculated with the UCM model fluid flow Reynolds number, dividing by the Reynolds of the Newtonian flow, $Reynolds\ ratio = \frac{Re_{UCM}}{Re_N}$. This evaluation is divided in two different studies, one having the Reynolds number of the UCM model fluid flow, fixed in a unitary value, $Re_{UCM} = 1$, and the other having the Newtonian fluid flow fixed in the same value, $Re_N = 1$. For better encapsulation of all the simulated flows, the following diagrams are presented including the unitary Reynolds ratio, common to both diagrams and previously shown.

6.5.1 Unitary Reynolds for the Newtonian fluid flow

The flow state for every Reynolds number ratio evaluated, with a fixed $Re_N = 1$ can be visualized in Figure 6.2. The steady/unsteady "boundary" is also highlighted, the real boundary is certainly not linear between the evaluated critical points so its purpose is merely figurative.

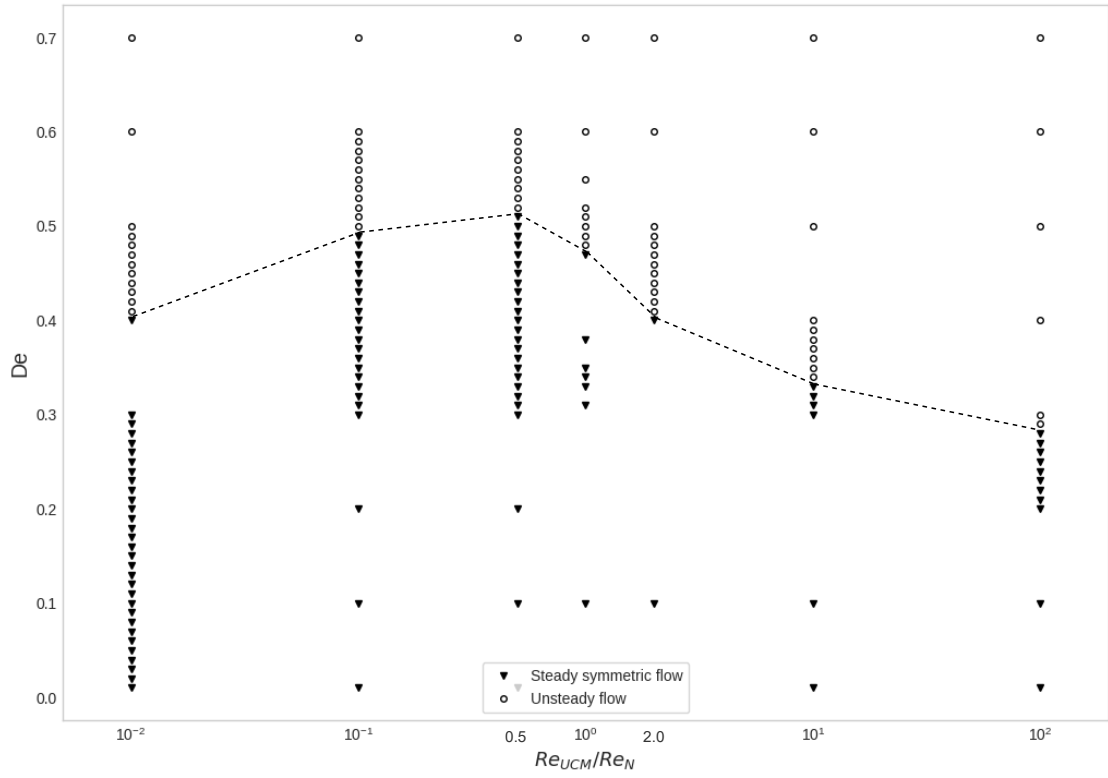


Figure 6.2: Flow state for each Reynolds number ratio, as a function of the Deborah number, with a fixed $Re_N = 1$ and $Ca = \infty$.

As represented, from the unitary ratio to higher values, the unsteady flow state is achieved for lower De values and when the ratio is less than one, the steadiness of the flow is maintained for higher De numbers. The conclusions on these results are presented after the $Re_{UCM} = 1$, as a more complete analysis can be made.

The interface representation for every Reynolds ratio with a fixed $Re_N = 1$ and $De = 0.1$ can be visualized in Figure 6.3.

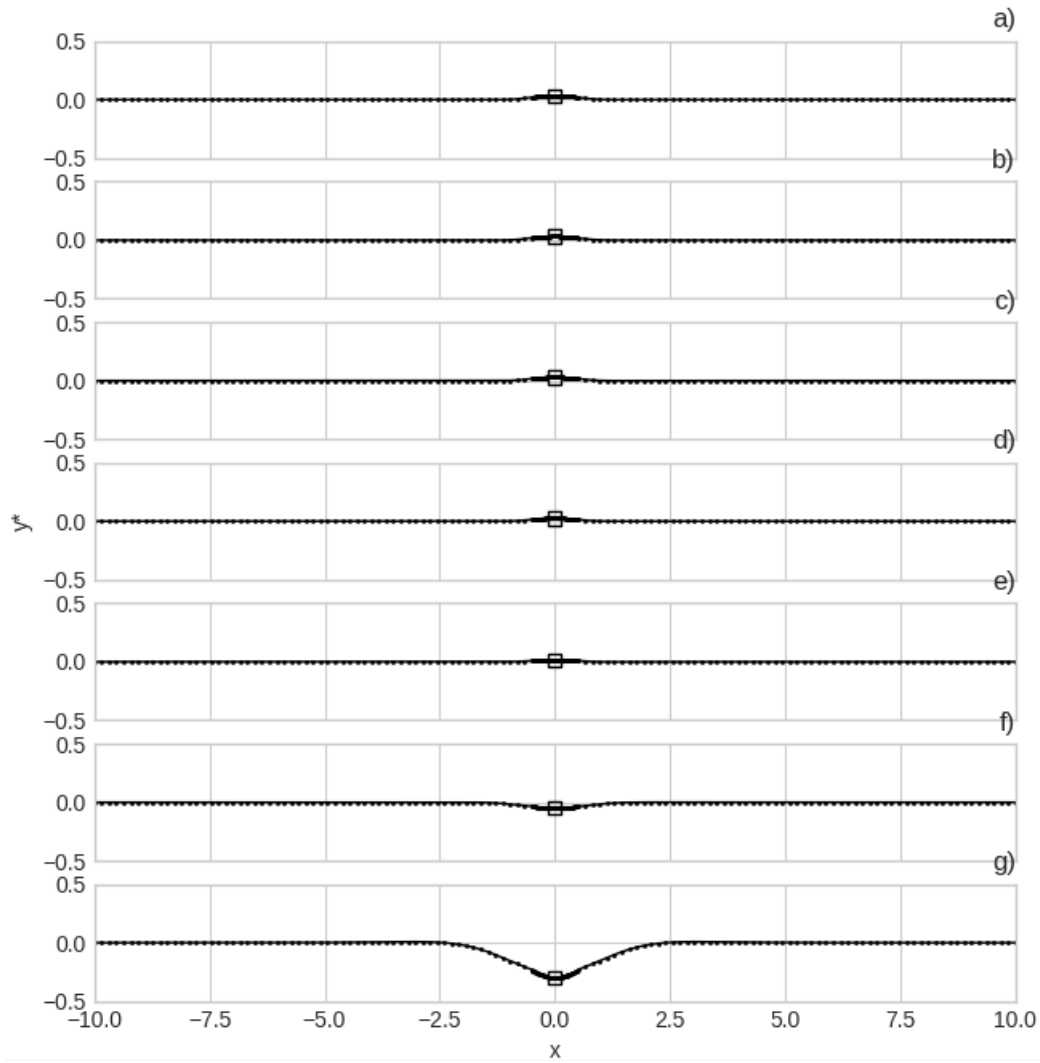


Figure 6.3: Interface representation and respective stagnation point coordinates (s.p.) when $De=0.1$, $Re_N = 1$ and $Re_{UCM}/Re_N =$: a) 0.01, s.p.: $(x,y^*)=(0.00,0.04)$; b) 0.1, s.p.: $(x,y^*)=(0.00,0.03)$; c) 0.5, s.p.: $(x,y^*)=(0.00,0.03)$; d) 1, s.p.: $(x,y^*)=(0.00,0.03)$; e) 2, s.p.: $(x,y^*)=(0.00,0.02)$; f) 10, s.p.: $(x,y^*)=(0.00,-0.05)$; g) 100, s.p.: $(x,y^*)=(0.00,-0.30)$. The square represents the stagnation point above which is the UCM fluid and below it, the Newtonian.

The simulated quantities used for this sub-chapter, as well as their local Weissenberg number, can be visualized in the following table.

Table 6.4: Local Weissenberg numbers for every simulated De and Reynolds number ratio, with $Re_N = 1$. * represents Unsteady flows, - represents not simulated pairs, ' represents steady asymmetry

Wi0		$Re_{UCM}/Re_N, Re_N = 1$						
		0.01	0.1	0.5	1.0	2.0	10	100
De	0.01	0.03	0.03	0.03	-	-	0.03	0.03
	0.02	0.06	-	-	-	-	-	-
	0.03	0.10	-	-	-	-	-	-
	0.04	0.13	-	-	-	-	-	-
	0.05	0.16	-	-	-	-	-	-
	0.06	0.20	-	-	-	-	-	-
	0.07	0.23	-	-	-	-	-	-
	0.08	0.26	-	-	-	-	-	-
	0.09	0.29	-	-	-	-	-	-
	0.10	0.32	0.31	0.31	0.32	0.32	0.34	0.33
	0.11	0.34	-	-	-	-	-	-
	0.12	0.37	-	-	-	-	-	-
	0.13	0.40	-	-	-	-	-	-
	0.14	0.43	-	-	-	-	-	-
	0.15	0.45	-	-	-	-	-	-
	0.16	0.48	-	-	-	-	-	-
	0.17	0.50	-	-	-	-	-	-
	0.18	0.52	-	-	-	-	-	-
	0.19	0.55	-	-	-	-	-	-
	0.20	0.56	0.56	0.55	-	-	-	0.60
	0.21	0.58	-	-	-	-	-	0.62
	0.22	0.60	-	-	-	-	-	0.65
	0.23	0.61	-	-	-	-	-	0.71
	0.24	0.62	-	-	-	-	-	0.69
	0.25	0.63	-	-	-	-	-	0.67
	0.26	0.64	-	-	-	-	-	0.65
	0.27	0.65	-	-	-	-	-	0.62
	0.28	0.66	-	-	-	-	-	0.59
	0.29	0.66	-	-	-	-	-	*
	0.30	0.67	0.67	0.69	-	-	0.74	*
	0.31	-	0.69	0.70	0.98	-	0.77	-
	0.32	-	0.70	0.72	-	-	0.81	-
	0.33	-	0.71	0.73	1.05	-	0.86	-
	0.34	-	0.72	0.75	1.08	-	*	-
	0.35	-	0.73	0.76	1.11	-	*	-
	0.36	-	0.74	0.78	-	-	*	-
	0.37	-	0.75	0.80	-	-	*	-
	0.38	-	0.76	0.83	1.20	-	*	-
	0.39	-	0.79	1.11	-	-	*	-

Wi₀		Re_{UCM}/Re_N, Re_N = 1						
		0.01	0.1	0.5	1.0	2.0	10	100
De	0.40	0.80	0.82	1.16	-	1.18	*	*
	0.41	*	0.85	1.20	-	*	-	-
	0.42	*	1.20	1.24	-	*	-	-
	0.43	*	1.25	1.28	-	*	-	-
	0.44	*	1.28	1.32	-	*	-	-
	0.45	*	1.32	1.36	-	*	-	-
	0.46	*	1.36	1.39	-	*	-	-
	0.47	*	1.40	1.43	1.55	*	-	-
	0.48	*	1.44	1.47	*	*	-	-
	0.49	*	1.48	1.51	*	*	-	-
	0.50	*	*	1.54	*	*	*	*
	0.51	-	*	1.58	*	-	-	-
	0.52	-	*	*	*	-	-	-
	0.53	-	*	*	-	-	-	-
	0.54	-	*	*	-	-	-	-
	0.55	-	*	*	*	-	-	-
	0.56	-	*	*	-	-	-	-
	0.57	-	*	*	-	-	-	-
	0.58	-	*	*	-	-	-	-
	0.59	-	*	*	-	-	-	-
	0.60	*	*	*	*	*	*	*
	0.70	*	*	*	*	*	*	*

From this table one can observe that the local Weissenberg number increase is related to the Deborah number increase. In terms of local Weissenberg number variation with the Reynolds ratio, for low De number flows the Wi_0 values seem to be very similar, unlike flows with higher De numbers where the Wi_0 values seem to be higher for near unitary ratios.

6.5.2 Unitary Reynolds for the UCM model fluid flow

The flow state for every Reynolds number ratio evaluated, with a fixed $Re_{UCM} = 1$ can be visualized in Figure 6.4. In this case a clear tendency is observed, unsteady flows are developed for lower Deborah numbers as the Reynolds ratio departs from its unitary value.

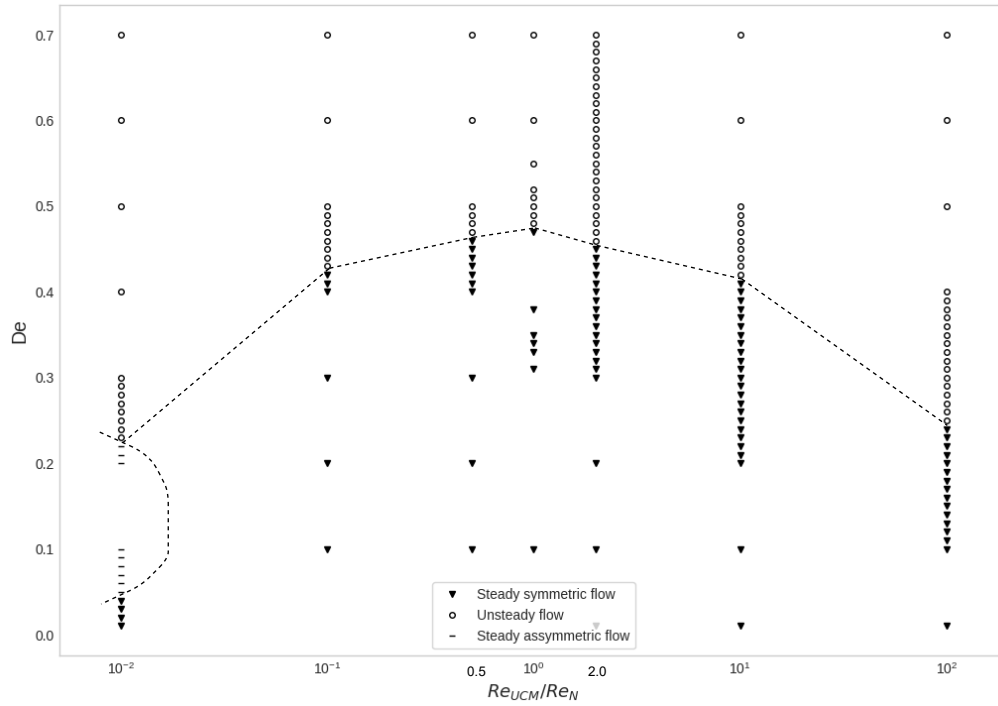


Figure 6.4: Flow state for each Reynolds ratio, as a function of the Deborah number, with a fixed $Re_{UCM} = 1$ and $Ca = \infty$.

Another interesting result is the steady state asymmetry, particular to the 0.01 ratio. In this range the Reynolds number of the Newtonian flow, is 100 times larger than its UCM model counterpart, and unlike every other case in this analysis, its stagnation point is not located in the $x \approx 0$ region. The interface representation for $De=0.1$ and all Reynolds ratios with the fixed $Re_{UCM} = 1$, as well as their stagnation point coordinates can be visualized in Figure 6.5.

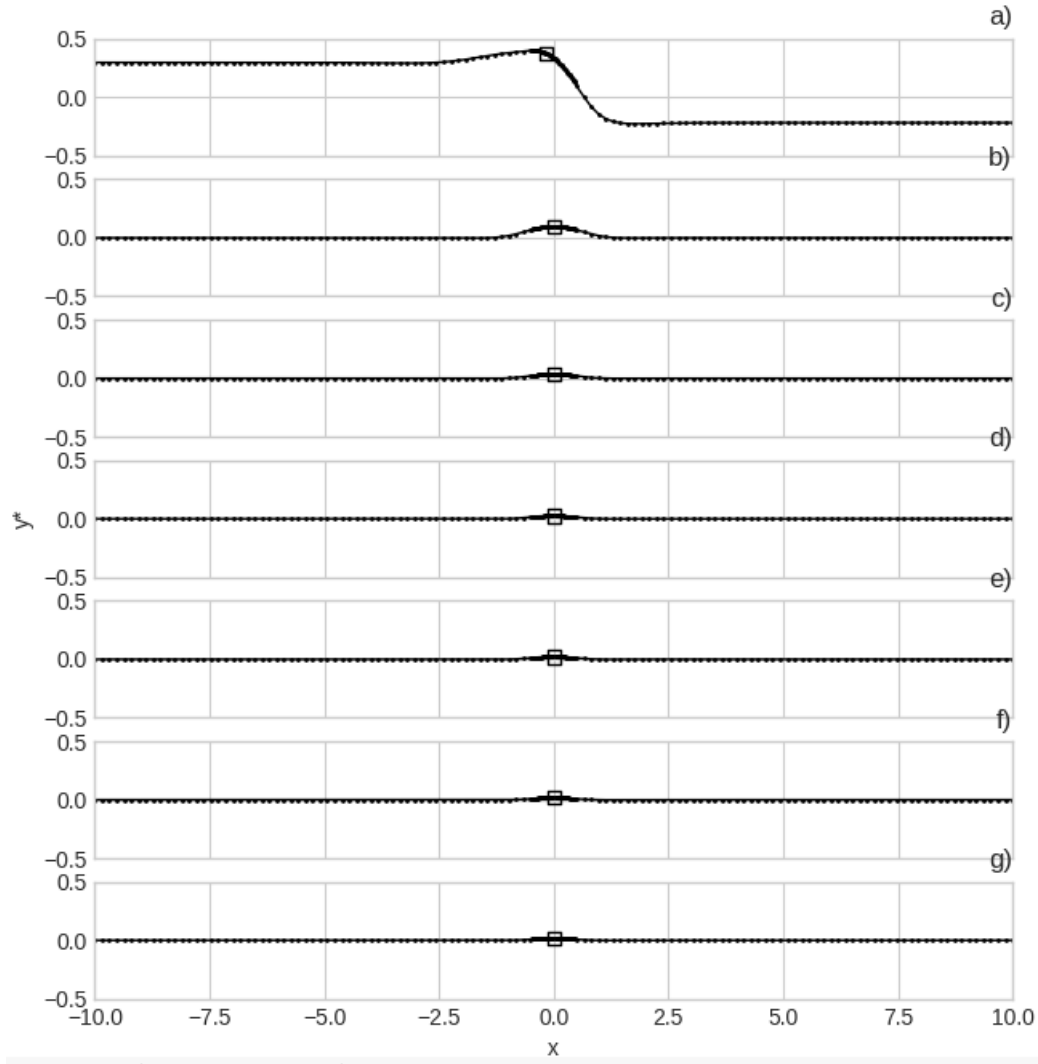


Figure 6.5: Interface representation and respective stagnation point coordinates (s.p.) when $De=0.1$, $Re_{UCM} = 1$ and $Re_{UCM}/Re_N =$: a) 0.01, s.p.: $(x,y^*)=(-0.16,0.37)$; b) 0.1, s.p.: $(x,y^*)=(0.00,0.10)$; c) 0.5, s.p.: $(x,y^*)=(0.00,0.04)$; d) 1, s.p.: $(x,y^*)=(0.00,0.03)$; e) 2, s.p.: $(x,y^*)=(0.00,0.02)$; f) 10, s.p.: $(x,y^*)=(0.00,0.02)$; g) 100, s.p.: $(x,y^*)=(0.00,0.20)$. The square represents the stagnation point above which is the UCM fluid and below it, the Newtonian.

One general flow behaviour that can be interpreted from Figures 6.3 and 6.5 is the stagnation point upper displacement for most of the evaluated Reynolds number ratios, including its unitary case. Common sense would suggest that the two flows would find balance in the $y=0$ position, where the stagnation point could be found. However since one of the fluids has elastic properties this is not necessarily the case and the interface representation reveals it to be slightly above this position ($y^* = 0.03$).

Even though for the a) case in Figure 6.5, the coordinates of the stagnation point reflect the flow asymmetry, this representation is not clear for that effect. Another method for the asymmetry of the flow evaluation is to represent, opposing velocity profiles, for example, at the entrance of

the cross-slot arms. In this case, if these match, the flow is considered symmetric otherwise the flow could either be unsteady or asymmetric in a steady state. This latter method is presented in Figure 6.6, for the previous a), d) and g) cases.

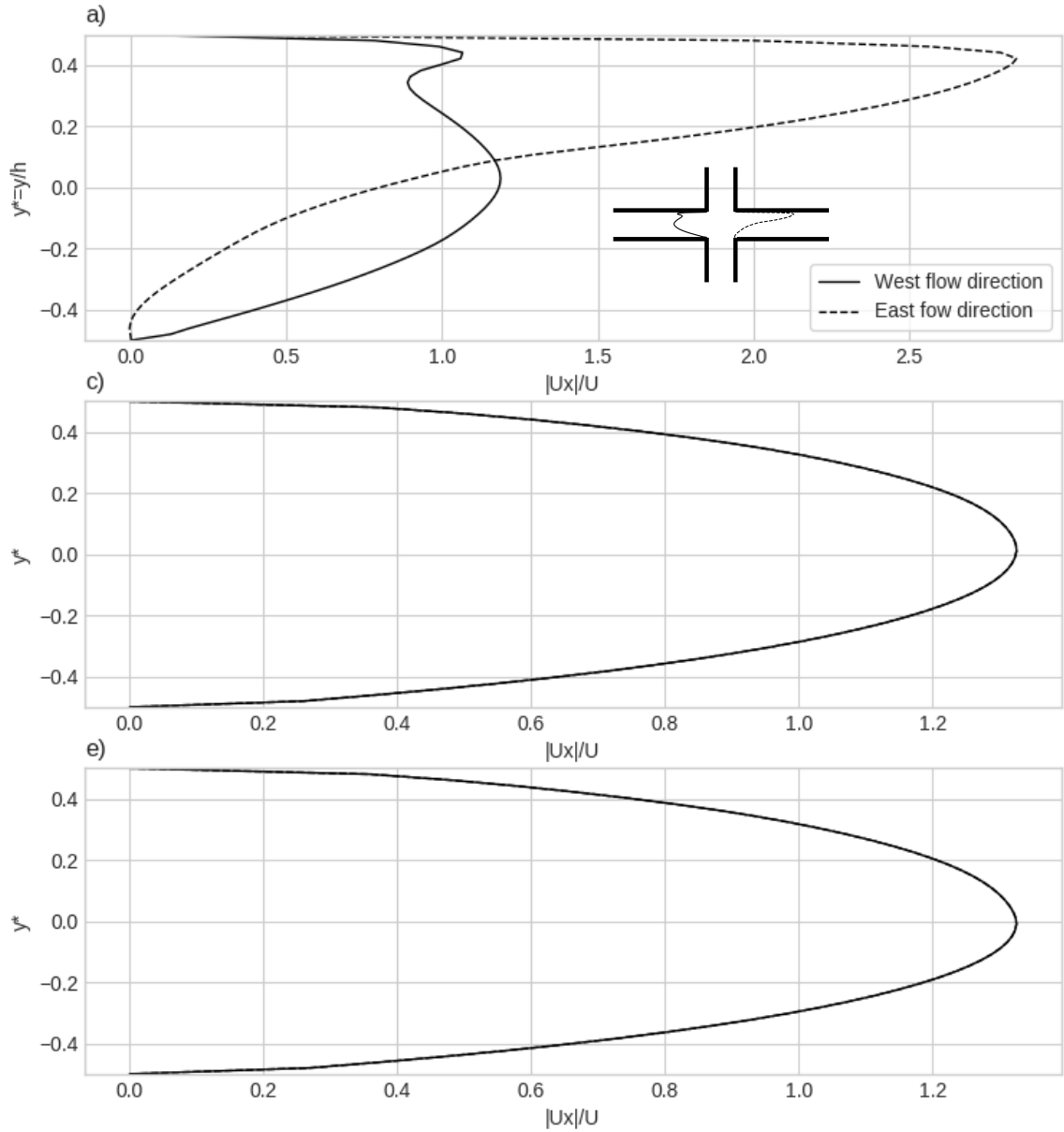


Figure 6.6: Normalized velocity profiles in absolute values for the cross-slot arms inlets at $x=0.5$ and $x=-0.5$, for the previous a), d) and g) cases. h represents the channels width, U represents the inlet average velocity of the flow.

In the d) and g) cases, these profiles coincide, as they should for symmetric flows.

By taking a look at the velocity profiles, slight distortions can be found such as mild x axis asymmetries. Note that the symmetry term used in this thesis refers to symmetry along the y axis, and this is verified by the coincidental velocity profiles at the channel outlets.

This sub-chapter simulated quantities, as well as their local Weissenberg number, can be visualized in the following table.

Table 6.5: Local Weissenberg numbers for every simulated De and Reynolds ratio, with $Re_{UCM} = 1$. * represents Unsteady flows, - represents not simulated pairs, ' represents steady asymmetry

Wi0		$Re_{UCM}/Re_N, Re_{UCM} = 1$						
		0.01	0.1	0.5	1.0	2.0	10	100
De	0.01	0.03	-	-	-	0.03	0.03	0.03
	0.02	0.06	-	-	-	-	-	-
	0.03	0.09	-	-	-	-	-	-
	0.04	0.13	-	-	-	-	-	-
	0.05	0.15'	-	-	-	-	-	-
	0.06	0.16'	-	-	-	-	-	-
	0.07	0.18'	-	-	-	-	-	-
	0.08	0.23'	-	-	-	-	-	-
	0.09	0.23'	-	-	-	-	-	-
	0.10	0.25'	0.31	0.32	0.32	0.32	0.32	0.32
	0.11	-	-	-	-	-	-	0.35
	0.12	-	-	-	-	-	-	0.38
	0.13	-	-	-	-	-	-	0.40
	0.14	-	-	-	-	-	-	0.43
	0.15	-	-	-	-	-	-	0.46
	0.16	-	-	-	-	-	-	0.48
	0.17	-	-	-	-	-	-	0.51
	0.18	-	-	-	-	-	-	0.53
	0.19	-	-	-	-	-	-	0.56
	0.20	0.47'	0.55	0.57	-	0.59	0.59	0.58
	0.21	0.49'	-	-	-	-	0.61	0.61
	0.22	0.49'	-	-	-	-	0.64	0.63
	0.23	*	-	-	-	-	0.67	0.66
	0.24	*	-	-	-	-	0.69	0.69
	0.25	*	-	-	-	-	0.72	*
	0.26	*	-	-	-	-	0.75	*
	0.27	*	-	-	-	-	0.78	*
	0.28	*	-	-	-	-	0.81	*
	0.29	*	-	-	-	-	0.83	*
	0.30	*	0.63	0.65	-	0.88	0.86	*
	0.31	-	-	-	0.98	0.91	0.89	*
	0.32	-	-	-	-	0.94	0.92	*
	0.33	-	-	-	1.05	0.97	0.95	*
	0.34	-	-	-	1.08	1.00	0.98	*
	0.35	-	-	-	1.11	1.04	1.00	*
	0.36	-	-	-	-	1.07	1.03	*
	0.37	-	-	-	-	1.11	1.06	*
	0.38	-	-	-	1.20	1.14	1.09	*
	0.39	-	-	-	-	1.18	1.12	*

Wi0		Re _{UCM} /Re _N , Re _{UCM} = 1						
		0.01	0.1	0.5	1.0	2.0	10	100
De	0.40	*	0.68	0.73	-	1.21	1.15	*
	0.41	-	0.69	0.75	-	1.24	1.19	-
	0.42	-	0.71	0.77	-	1.28	*	-
	0.43	-	*	0.80	-	1.31	*	-
	0.44	-	*	0.83	-	1.35	*	-
	0.45	-	*	0.86	-	1.38	*	-
	0.46	-	*	0.89	-	*	*	-
	0.47	-	*	*	1.55	*	*	-
	0.48	-	*	*	*	*	-	-
	0.49	-	*	*	*	*	*	-
	0.50	*	*	*	*	*	*	*
	0.51	-	-	-	*	*	-	-
	0.52	-	-	-	*	*	-	-
	0.53	-	-	-	*	*	-	-
	0.54	-	-	-	-	*	-	-
	0.55	-	-	-	*	*	-	-
	0.56	-	-	-	-	*	-	-
	0.57	-	-	-	-	*	-	-
	0.58	-	-	-	-	*	-	-
	0.59	-	-	-	-	*	-	-
	0.60	*	*	*	*	*	*	*
	0.61	-	-	-	-	*	-	-
	0.62	-	-	-	-	*	-	-
	0.63	-	-	-	-	*	-	-
	0.64	-	-	-	-	*	-	-
	0.65	-	-	-	-	*	-	-
	0.66	-	-	-	-	*	-	-
	0.67	-	-	-	-	*	-	-
	0.68	-	-	-	-	*	-	-
	0.69	-	-	-	-	*	-	-
	0.70	*	*	*	*	*	*	*

6.5.3 Steady asymmetric flow mapping

In this sub-chapter, the mapping of the steady asymmetric flow region for the Reynolds ratio between 0.01 and 0.1 with the fixed $Re_{UCM} = 1$ is presented. Since the extreme Deborah numbers for these two ratios are 0.04 (lowest) and 0.43 (highest), it is in this range that the simulations will be conducted, in this case from 0.01 to 0.45. It was arbitrated to evaluate three different Reynolds ratios (0.0562, 0.0316, 0.0178) obtained by having the respective Newtonian Reynolds numbers (17.783, 31.622, 56.234) corresponding to their specific mass, since the average velocity, viscosity and characteristic length are unitary. The results can be visualized in Figure 6.7 and their local Weissenberg values can be consulted in Table 6.6.

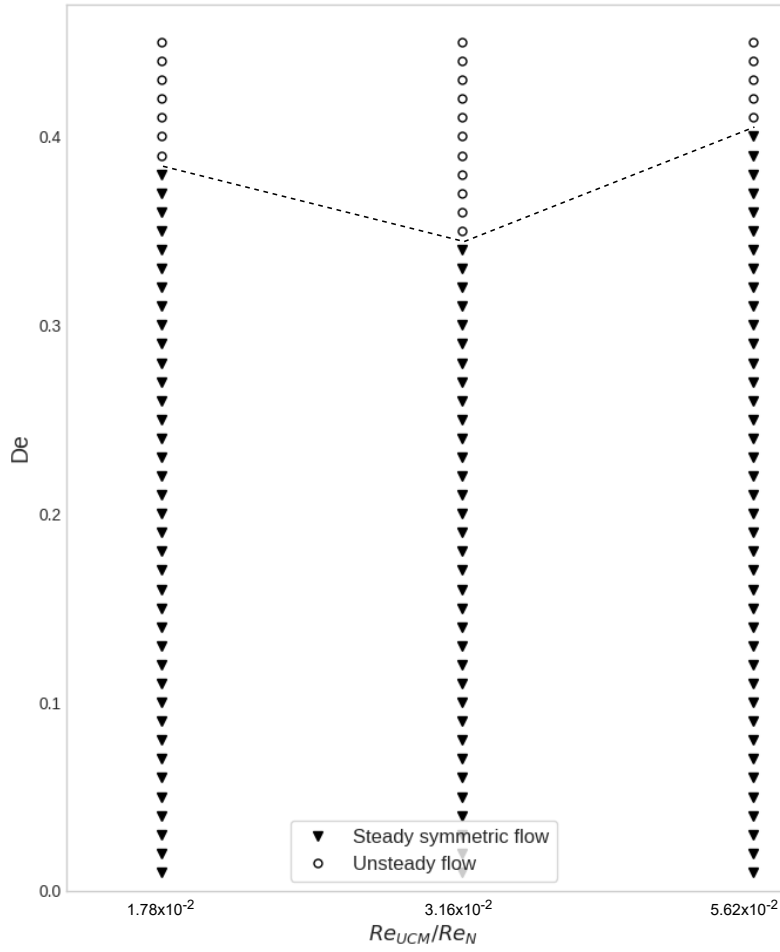


Figure 6.7: $De - Re_{UCM}/Re_N$ phase diagrams in the range 0.018-0.056 for $Re_{UCM} = 1$ and $Ca = \infty$.

Table 6.6: Local Weissenberg numbers for every simulated De and Reynolds number ratio, with $Re_{UCM} = 1$. * represents Unsteady flows.

Wi0		Re _{UCM} /Re _N , Re _{UCM} = 1		
		0.018	0.032	0.056
De	0.01	0.03	0.03	0.03
	0.02	0.06	0.06	0.06
	0.03	0.10	0.09	0.09
	0.04	0.12	0.12	0.13
	0.05	0.15	0.16	0.16
	0.06	0.17	0.18	0.19
	0.07	0.21	0.21	0.21
	0.08	0.23	0.23	0.24
	0.09	0.25	0.25	0.26
	0.10	0.26	0.30	0.29

Wi0		Re_{UCM}/Re_N, Re_{UCM} = 1		
		0.018	0.032	0.056
De	0.11	0.34	0.32	0.33
	0.12	0.36	0.35	0.36
	0.13	0.38	0.37	0.38
	0.14	0.40	0.39	0.41
	0.15	0.42	0.41	0.43
	0.16	0.43	0.43	0.45
	0.17	0.45	0.45	0.48
	0.18	0.46	0.47	0.50
	0.19	0.47	0.49	0.52
	0.20	0.47	0.50	0.54
	0.21	0.48	0.52	0.57
	0.22	0.48	0.54	0.58
	0.23	0.48	0.56	0.60
	0.24	0.49	0.59	0.60
	0.25	0.49	0.61	0.60
	0.26	0.49	0.63	0.59
	0.27	0.49	0.66	0.58
	0.28	0.50	0.69	0.59
	0.29	0.52	0.71	0.59
	0.30	0.53	0.74	0.60
	0.31	0.55	0.77	0.61
	0.32	0.58	0.79	0.77
	0.33	0.60	0.81	0.81
	0.34	0.63	0.83	0.87
	0.35	0.66	*	0.91
	0.36	0.69	*	0.96
	0.37	0.71	*	1.01
	0.38	0.73	*	1.06
	0.39	*	*	1.10
	0.40	*	*	1.13
	0.41	*	*	*
	0.42	*	*	*
	0.43	*	*	*
	0.44	*	*	*
	0.45	*	*	*

The results explored in this sub-section lack the presence of the initially expected steady asymmetric flow. Plus the critical Deborah numbers of the ratios here analysed show an opposite tendency comparing to the previous values (see Figure 6.4), where the critical Deborah number tends to be higher as a unitary Reynolds ratio is approached. These results show that a more in depth exploration is required in order to have a complete behavioral map of these types of flows, however they demonstrate that steady asymmetry is a rare state to achieve in UCM/Newtonian fluid flows in this chapter conditions, only appearing in flows of $Re_N = 100$ and $Re_{UCM} = 1$.

Chapter 7

Interfacial flow between multiphase UCM fluids

7.1 Introduction

In this chapter the Upper-convected Maxwell model fluid flow in the cross-slot geometry is studied. For a single phase flow this study has been extensively explored, however the two-phase fluid flow has not. Since this is one of the main focuses of this dissertation, in the present chapter, different fluid pairs described by the UCM model are introduced in the opposing channels of the cross-slot setting.

Every simulation result presented in this chapter was performed on a non-uniform mesh geometry due to the use of the *simpleGrading* tool, previously described.

To better encapsulate the main focus physics subjects of this chapter (viscosity and elasticity), every flow presented in the present chapter have a significantly low Reynolds number of $Re=0.01$ in order to reduce inertial effects.

7.2 Flow description

In order to explore an interesting range of characteristics, the starting point of all simulations is the single phase, non-inertial, UCM model. The critical Deborah number in the cross-slot geometry, 0.31. Having two different fluids, the idea was to interpret the changes both Deborah number variations have on the flow physics, note that having two different phases described by the UCM model implies having two De numbers. For the upper channel fluid flow five Deborah number values were fixed (0.29, 0.30, 0.31, 0.32 and 0.33) while altering the Deborah number of the fluid entering from the lower channel.

In all simulations presented in this chapter there is no surface tension in the interface of the fluids

$(Ca = \infty)$.

The utilized nomenclature for this chapter, namely to distinguish the present fluids refers to the upper channel fluid flow as UCM_1 , and to the lower channel fluid flow as UCM_2 . The fixed Deborah number flows were therefore, always for the UCM_1 fluid (fixed De_1).

7.3 Results

The obtained results show good correspondence with the single phase critical value of $De=0.31$ [6]. In this work simulations the $De=0.31$ pair is symmetric but the 0.01 Deborah number increment in just one of the fluids, is enough to have the asymmetric steady state. The results also agree in local Weissenberg values and flow behaviour when a Deborah number pair is simulated twice (switching fluids between inlets) as expected since the gravity acceleration is not considered.

The complete results of the simulations can be visualized in Figure 7.1 and their local Weissenberg numbers can be consulted in Table 7.1.

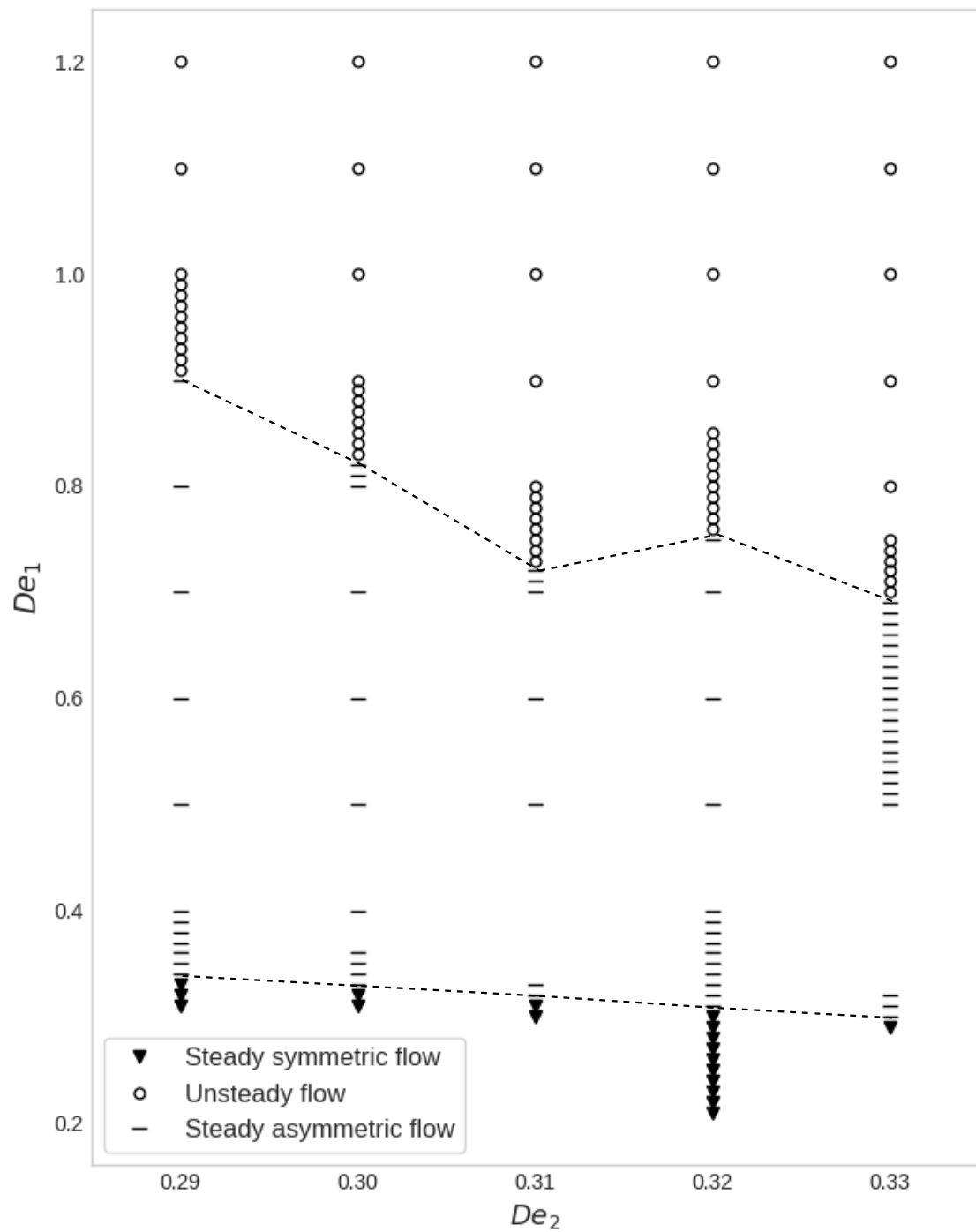


Figure 7.1: $De_1 - De_2$ phase diagram for UCM/UCM flow at $Re_1 = 0.01$, $Re_2 = 0.01$ and $Ca = \infty$.

[illegible]

By analysing Table 7.1 and comparing the local Weissenberg values with the ones presented in Chapter 6 two different tendencies can be inferred. In the UCM/Newtonian fluids flow, a general increase of the local Weissenberg value can be observed when the Deborah number is increased, unlike for UCM/UCM flows, where higher Wi_0 values can be found in the middle of the Deborah number range, presenting also more general variations.

7.4 Near Symmetry breaking flows study

It was observed that drifting from symmetry breaking critical values the stagnation point is no longer in the absolute center of the geometry ($x=0; y=0$), as it is in every steady symmetric flow for the UCM/UCM case, it is just near that region. This fact could justify the variations of the local Weissenberg calculations that require this position, and are "interpolation-free" when the stagnation point is in the $(x,y)=(0,0)$ coordinates. Figure 7.2 demonstrates this tendency, showing the stagnation point distance to the center of the geometry, or the stagnation point displacement, compared to the Deborah number difference between the two fluids present in every simulated asymmetric steady flow of this chapter.

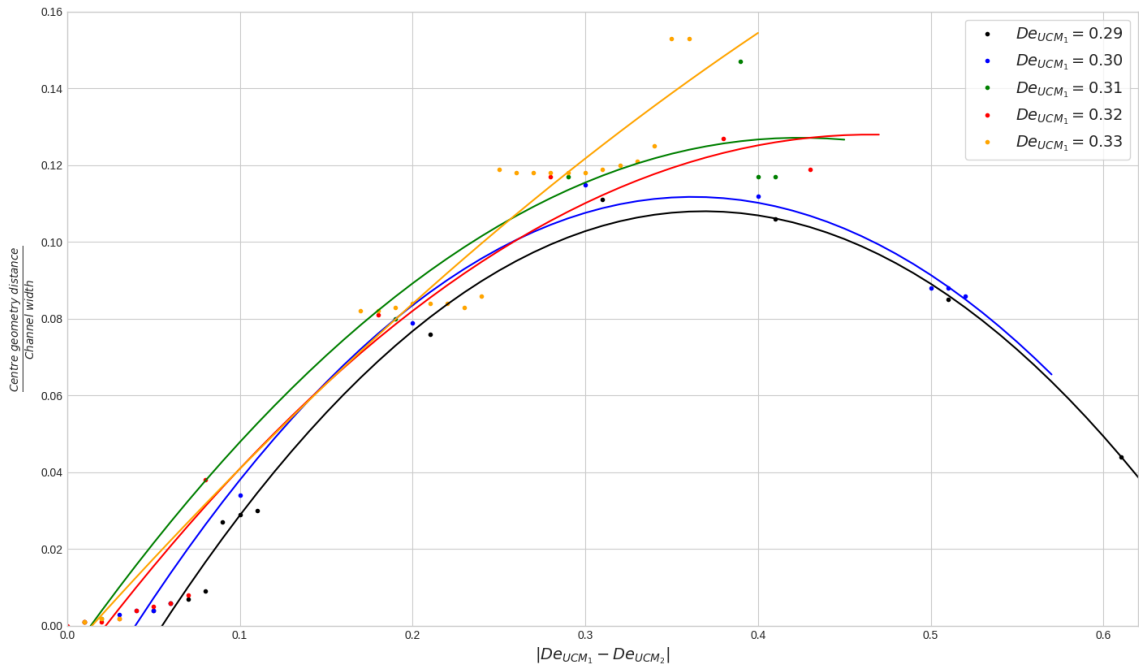


Figure 7.2: Stagnation point normalized distance from the geometry center, of every simulated steady asymmetric flow according to the $|De_{UCM1} - De_{UCM2}|$ difference.

Note that Figure 7.2 could also justify the absence of a linear increment on the local Weissenberg numbers as the Deborah number of the flow increases, since different placed stagnation points change local flow kinematics and could result in more abrupt velocity gradients. Therefore

a local Weissenberg number increase at the Deborah number mid-range could result from the stagnation point displacement, and, its decrease near unsteadiness also follows this trend since this displacement is lower.

The local Weissenberg decrease is not observed for the higher $De_{UCM_1} = 0.33$ values simulated since the stagnation point displacement does not decrease as well, but this can be observed in the $De_{UCM_1} = 0.29$ flow local Weissenberg values. In either cases Wi_0 values tend to accompany the stagnation point displacement tendency.

In terms of flow behaviour, this displacement results in a completely asymmetric flow, unlike near critical flows, where one outlet channel interface is the inverse function of the other meaning that interfaces coincide when a symmetry along the x and y axis is applied.

These facts suggest that the evaluation of flow behaviour with the present settings, is even more complex and uncertain at higher Deborah numbers, and, it should be studied with more data and higher precision. Taking this into account, near symmetry breaking flows are the focus of this subchapter since they represent the adequate analysis in order to better understand the flow physics of the present chapter.

For that end, the fixed $De_{UCM_1} = 0.32$ together with five near critical flows are considered. By analysing the interfaces of these flows (Figure 7.3) its possible to observe that, in the steady asymmetric cases, the degree of flow asymmetry gets increasingly stronger within each De_{UCM_2} increment. This increase in asymmetry is mainly present in the near critical flows, where the position of the stagnation point is at the centre of the geometry.

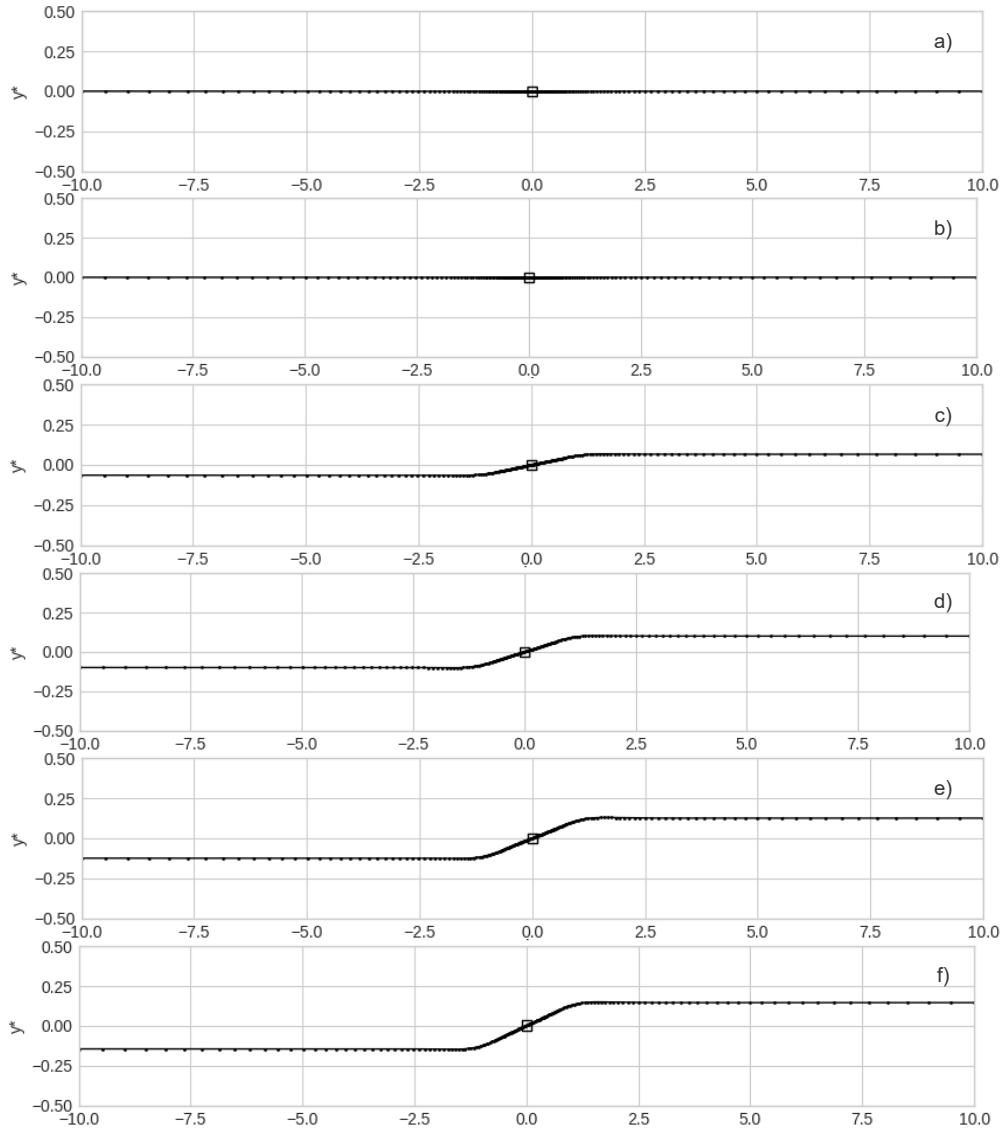


Figure 7.3: Interface representation of the near critical and critical flow conditions. a) $De_{UCM_2} = 0.29$, b) $De_{UCM_2} = 0.30$, c) $De_{UCM_2} = 0.31$, d) $De_{UCM_2} = 0.32$, e) $De_{UCM_2} = 0.33$, f) $De_{UCM_2} = 0.34$

The velocity profiles at the entrance of the outlet channels, for the same five near critical values, are present in Figure 7.4, where a better visualization of this increase in flow asymmetry, and confirmation of the inverse function between the interfaces of opposing outlets, is possible. Evidently, these velocity profiles have opposing directions, but inverting them along the y axis is better suited to demonstrate their correspondence if we were to invert one of them along the x axis.

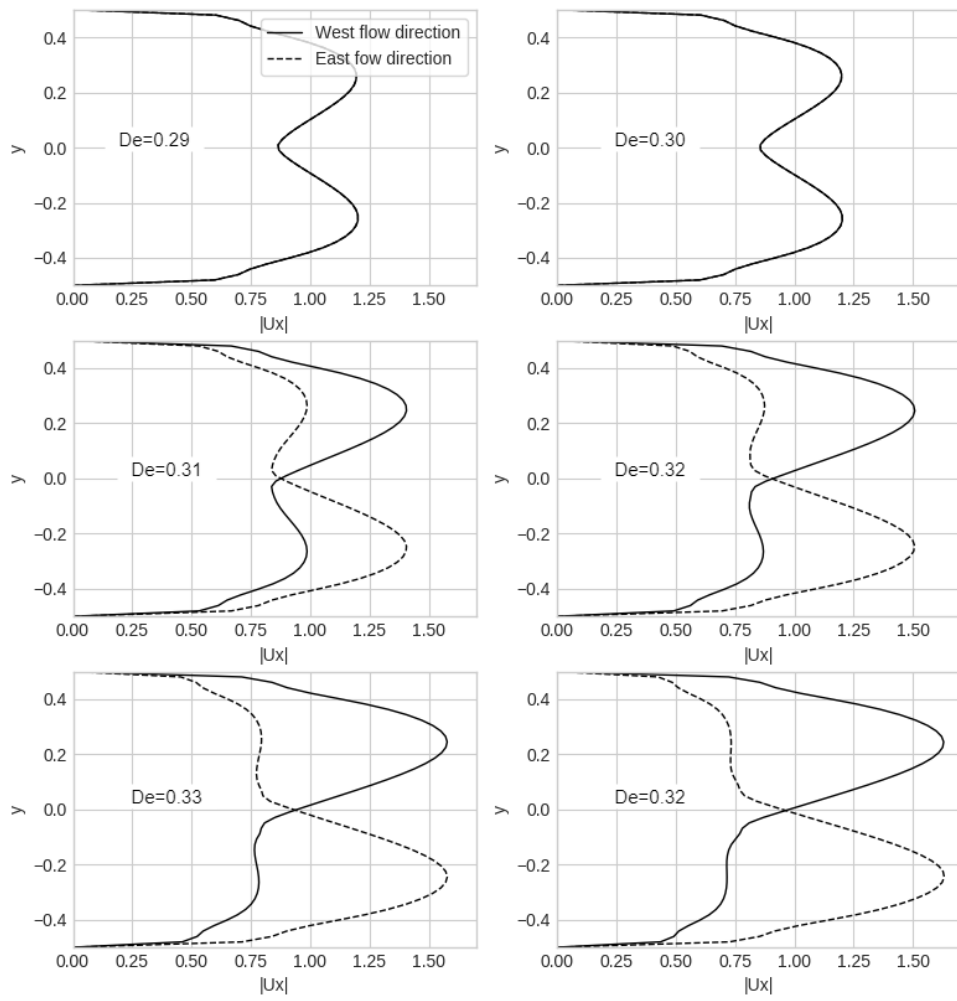


Figure 7.4: Velocity profile representation, for a fixed Deborah of 0.32 in the upper flow.

The stress fields of the analysed cases can also be visualized in Figure 7.5, where an asymmetry in the high stress center field is also worthy of attention.

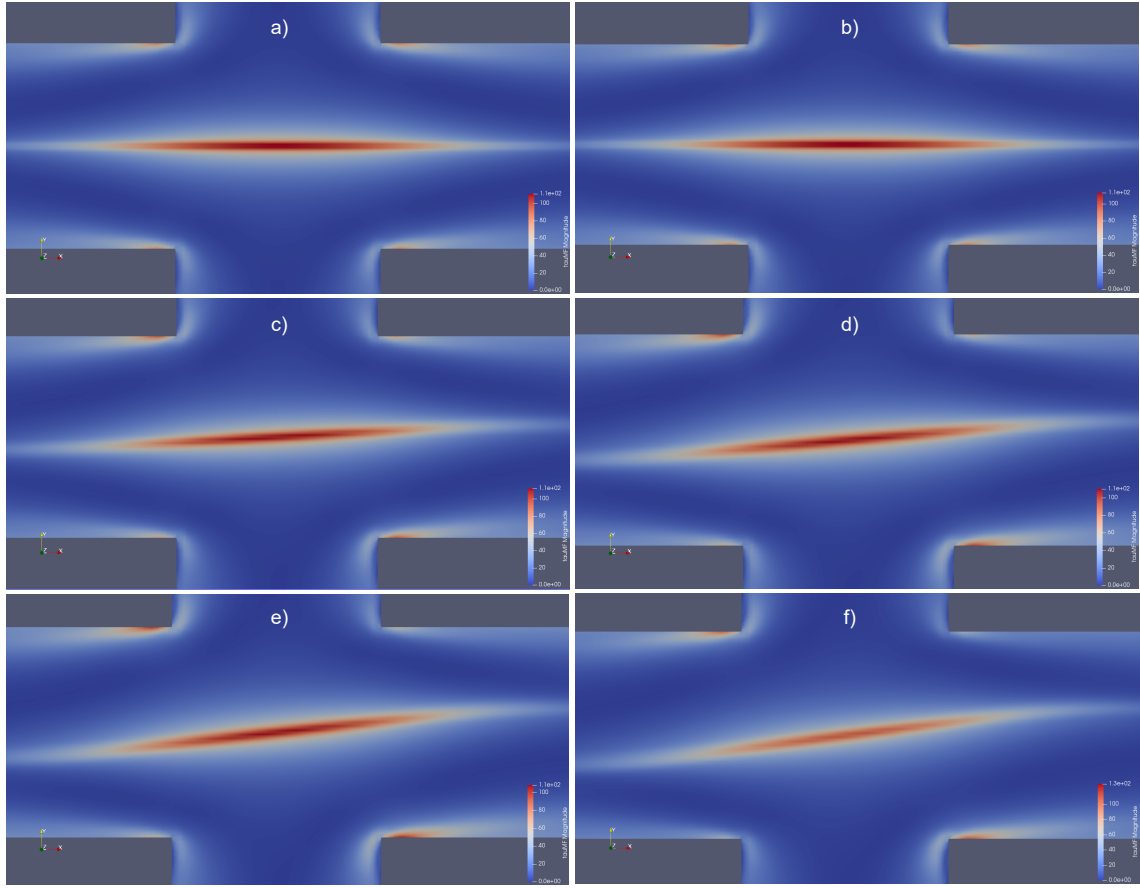


Figure 7.5: Polymeric extra-stress tensor magnitude in the center region of the cross-slot geometry. a) $De_{UM_2} = 0.29$, b) $De_{UM_2} = 0.30$, c) $De_{UM_2} = 0.31$, d) $De_{UM_2} = 0.32$, e) $De_{UM_2} = 0.33$, f) $De_{UM_2} = 0.34$

The stress increase due to the De_{UM_2} increase is further analysed in Table 7.2, where the upper and lower fluids stress magnitudes as well as the polymeric extra-stress tensor magnitude on the numeric cell containing the stagnation point are presented.

Table 7.2: Stagnation point stress stress magnitude contributions. $De_1 = 0.32$, $Re=0.01$, $Ca = \infty$ and the stagnation point is located at the center of the geometry $((x,y)=(0,0))$.

		Stagnation point stresses [Pa]		
		Upper fluid contribution	Polymeric extra-stress	Lower fluid contribution
De_2	0.29	123.3	110.0	93.8
	0.30	119.6	112.3	104.3
	0.31	106.9	106.9	106.9
	0.32	97.6	104.6	111.1
	0.33	89.4	102.8	114.4
	0.34	81.8	101.6	117.4

These results represent the maximum stresses one numerical cell encapsulates in every flow,

and a few tendencies can be observed. Polymeric extra-stress is relieved when the steady asymmetry state is reached, the upper fluid contribution on the stress gets lower with the De_{UCM_2} increase and the lower fluid contribution is higher with the increase of this quantity. Note that the UCM_2 fluid relaxation time directly follows its stress contribution suggesting that a higher relaxation time could implicate stress accumulation.

Chapter 8

Interfacial flow between multiphase Oldroyd-B fluids

8.1 Introduction

In this chapter the interfacial flow between multiphase Oldroyd-B model fluids is studied. This analysis poses as a continuation of the UCM/UCM flows since the Oldroyd-B model can be regarded as a generalization of the UCM model with the only difference between them being the consideration of the solvent viscosity in the Oldroyd-B case (see Chapter 3 for more details). This fluid flow has been extensively studied, however its multiphase configuration in a cross-slot geometry has not.

Every simulation result presented in this chapter was performed on a non-uniform mesh geometry due to the use of the *simpleGrading* tool, previously described.

To better encapsulate the main focus physics subjects of this chapter (viscosity and elasticity), every flow presented in the present chapter have a significantly low Reynolds number of $Re=0.01$ to avoid inertial contributions.

8.2 Flow description

The performed simulations aimed to evaluate the different Deborah number effects (De_1 and De_2), as previously done for the UCM case, and to study the influence of the ratio between the solvent viscosity and the total viscosity $\beta = \eta_s/\eta_0$, with $\eta_0 = \eta_s + \eta_p$. In all simulations presented in this chapter there is no surface tension in the interface of the fluids ($Ca = \infty$).

The utilized nomenclature for this chapter, refers to the upper channels fluid as OdB_1 and the lower channels fluid as OdB_2 . The β value for OdB_1 is referred to as β_1 and for OdB_2 , β_2 .

This analysis was set to part from a $\beta = 0$ value so it also includes a UCM/Oldroyd-B multiphase flow. Note that a null beta represents an UCM fluid.

8.3 Results

Table 8.1: Local Weissenberg values for all simulated flows with $De_1 = 0.29$, with $Ca=\infty$ and $Re=0.01$. Dist. stands for the stagnation point geometry center distance.

[illegible]

Table 8.3: Local Weissenberg values for all simulated flows with $De_1 = 0.31$, with $Ca=\infty$ and $Re=0.01$. * represents unsteady flows, Dist. stands for the stagnation point geometry center distance.

			De ₂										
			0.28	0.29	0.30	0.31	0.32	0.33	0.34	0.35	0.36	0.37	
De ₁ = 0.31	β ₁ = 0	β ₂ = 0.1											
		Wi ₀	0.62	0.60	0.59	0.59	0.60	0.60	0.60	0.60	0.60	*	
		Dist.	0.009	0.010	0.011	0.012	0.013	0.013	0.014	0.015	0.015	*	
		β ₂ = 0.5											
		Wi ₀	0.63	0.63	0.63	0.63	0.63	0.62	0.62	0.61	0.60	*	
		Dist.	0.079	0.079	0.080	0.080	0.080	0.081	0.081	0.082	0.082	*	
		β ₂ = 0.9											
		Wi ₀	*	*	*	*	*	*	*	*	*	*	
	Dist.	*	*	*	*	*	*	*	*	*	*		
	β ₁ = 0.1	β ₂ = 0.1											
		Wi ₀	0.59	0.59	0.60	0.60	0.61	0.60	0.60	0.60	0.59	0.59	
		Dist.	0.002	0.001	0.001	0.003	0.001	0.001	0.002	0.002	0.003	0.004	
		β ₂ = 0.5											
		Wi ₀	0.68	0.68	0.67	0.66	0.66	0.66	0.65	0.65	0.69	0.70	
		Dist.	0.065	0.065	0.066	0.066	0.067	0.067	0.068	0.068	0.069	0.069	
		β ₂ = 0.9											
		Wi ₀	0.59	0.59	0.60	0.61	0.61	0.62	0.62	0.63	0.63	0.64	
		Dist.	0.243	0.243	0.243	0.243	0.244	0.244	0.244	0.244	0.244	0.244	
		β ₁ = 0.5	β ₂ = 0.5										
			Wi ₀	0.73	0.72	0.71	0.70	0.73	0.74	0.76	0.77	0.78	0.79
			Dist.	0.000	0.000	0.000	0.001	0.000	0.000	0.000	0.000	0.000	0.000
	β ₂ = 0.9												
	Wi ₀		0.78	0.78	0.78	0.78	0.79	0.79	0.79	0.79	0.79	0.79	
	Dist.		0.160	0.160	0.160	0.160	0.160	0.160	0.160	0.160	0.160	0.160	
	β ₁ = 0.9		β ₂ = 0.9										
	Wi ₀		0.92	0.93	0.94	0.95	0.97	0.99	1.00	1.01	1.03	1.04	
	Dist.	0.000	0.000	0.000	0.002	0.000	0.000	0.000	0.000	0.000	0.000		

Table 8.4: Local Weissenberg values for all simulated flows with $De_1 = 0.32$, with $Ca=\infty$ and $Re=0.01$. * represents unsteady flows, ' stands for steady asymmetric flow, Dist. stands for the stagnation point geometry center distance.

			De_2									
			0.28	0.29	0.30	0.31	0.32	0.33	0.34	0.35	0.36	0.37
$De_1 = 0.32$	$\beta_1 = 0$	$\beta_2 = 0.1$										
		Wi_0	0.63	0.62	0.58	0.59	0.60	0.60	0.60	0.61	0.63'	0.63'
		Dist.	0.009	0.009	0.010	0.011	0.012	0.013	0.013	0.014	0.019'	0.020'
		$\beta_2 = 0.5$										
		Wi_0	0.64	0.64	0.64	0.64	0.63	0.63	0.62	0.61	0.61	0.60
		Dist.	0.078	0.079	0.080	0.080	0.080	0.081	0.081	0.081	0.082	0.082
		$\beta_2 = 0.9$										
		Wi_0	*	*	*	*	*	*	*	*	*	*
		Dist.	*	*	*	*	*	*	*	*	*	*
	$\beta_1 = 0.1$	$\beta_2 = 0.1$										
		Wi_0	0.58	0.59	0.59	0.61	0.60	0.62	0.60	0.60	0.60	0.59
		Dist.	-0.002	-0.001	-0.001	-0.001	0.000	0.001	0.001	0.002	0.002	0.003
		$\beta_2 = 0.5$										
		Wi_0	0.69	0.69	0.68	0.68	0.67	0.67	0.66	0.66	0.66	0.69
		Dist.	0.065	0.065	0.066	0.066	0.066	0.067	0.067	0.068	0.068	0.069
		$\beta_2 = 0.9$										
		Wi_0	0.60	0.60	0.61	0.61	0.62	0.62	0.63	0.63	0.64	0.64
		Dist.	0.244	0.244	0.244	0.244	0.244	0.244	0.244	0.244	0.245	0.245
	$\beta_1 = 0.5$	$\beta_2 = 0.5$										
		Wi_0	0.74	0.73	0.73	0.73	0.73	0.73	0.75	0.76	0.77	0.79
		Dist.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		$\beta_2 = 0.9$										
		Wi_0	0.79	0.80	0.80	0.80	0.81	0.81	0.81	0.81	0.81	0.81
		Dist.	0.160	0.160	0.160	0.160	0.160	0.160	0.160	0.160	0.160	0.160
	$\beta_1 = 0.9$	$\beta_2 = 0.9$										
		Wi_0	0.93	0.95	0.96	0.97	0.98	1.00	1.01	1.03	1.04	1.05
		Dist.	0.000	0.000	0.000	0.000	0.002	0.000	0.000	0.000	0.000	0.000

Table 8.5: Local Weissenberg values for all simulated flows with $De_1 = 0.33$, with $Ca=\infty$ and $Re=0.01$. * represents unsteady flows, ' stands for steady asymmetric flow, Dist. stands for the stagnation point geometry center distance.

			De_2									
			0.28	0.29	0.30	0.31	0.32	0.33	0.34	0.35	0.36	0.37
$De_1 = 0.33$	$\beta_1 = 0$	$\beta_2 = 0.1$										
		Wi_0	0.64	0.63	0.63	0.58	0.59	0.60	0.60	0.63'	0.61'	0.59'
		Dist.	0.008	0.009	0.009	0.010	0.011	0.012	0.013	0.016'	0.018'	0.019'
		$\beta_2 = 0.5$										
		Wi_0	0.64	0.64	*	*	*	*	*	*	*	*
		Dist.	0.079	0.079	*	*	*	*	*	*	*	*
		$\beta_2 = 0.9$										
		Wi_0	*	*	*	*	*	*	*	*	*	*
		Dist.	*	*	*	*	*	*	*	*	*	*
	$\beta_1 = 0.1$	$\beta_2 = 0.1$										
		Wi_0	0.58	0.59	0.59	0.60	0.61	0.62	0.62	0.60	0.60	0.60
		Dist.	-0.003	-0.002	-0.002	-0.001	-0.001	0.000	0.001	0.001	0.002	0.002
		$\beta_2 = 0.5$										
		Wi_0	0.71	0.70	0.70	0.69	0.69	0.68	0.67	0.67	0.67	0.66
		Dist.	0.065	0.065	0.065	0.066	0.066	0.067	0.067	0.068	0.068	0.068
		$\beta_2 = 0.9$										
		Wi_0	0.60	0.60	0.61	0.61	0.62	0.62	0.63	0.63	0.64	0.64
		Dist.	0.244	0.244	0.244	0.244	0.245	0.245	0.245	0.245	0.245	0.245
	$\beta_1 = 0.5$	$\beta_2 = 0.5$										
		Wi_0	0.75	0.75	0.74	0.74	0.73	0.71	0.74	0.75	0.77	0.78
		Dist.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		$\beta_2 = 0.9$										
		Wi_0	0.81	0.82	0.82	0.82	0.82	0.83	0.83	0.83	0.83	0.83
		Dist.	0.160	0.160	0.160	0.160	0.160	0.160	0.160	0.160	0.160	0.160
	$\beta_1 = 0.9$	$\beta_2 = 0.9$										
		Wi_0	0.95	0.96	0.97	0.99	1.00	1.01	1.03	1.04	1.05	1.06
		Dist.	0.000	0.000	0.000	0.000	0.000	0.002	0.000	0.000	0.000	0.000

By evaluating these results a few inferences can be made. Steady asymmetric flows as well as unsteady flows are only present where $\beta = 0$, meaning that in Oldroyd-B/Oldroyd-B flows, for the evaluated variables, the flow is always steady symmetric unlike the considered UCM/Oldroyd-B cases. This reveals a stabilizing nature of the added solvent but unfortunately the evaluation of

flow steadiness limits was out of our reach due to the limited time resources this thesis predisposes. Another observation can be made in terms of the influence of the Deborah number of the flow. Either the De_1 and the De_2 variations show little flow influence in terms of flow steadiness, local Weissenberg values and stagnation point displacement. These results are also interesting in terms of the stagnation point position, that generally revealed to be in the center of the geometry for flows with $\beta_1/\beta_2 = 1$ and increasingly distant as this ratio departs from an unitary value. This behaviour can be observed in Figure 8.1 where the interface representation for the three different values of β_2 is showed, for the flows with $De_1 = 0.33$, $De_2 = 0.33$ and $\beta_1 = 0.1$.

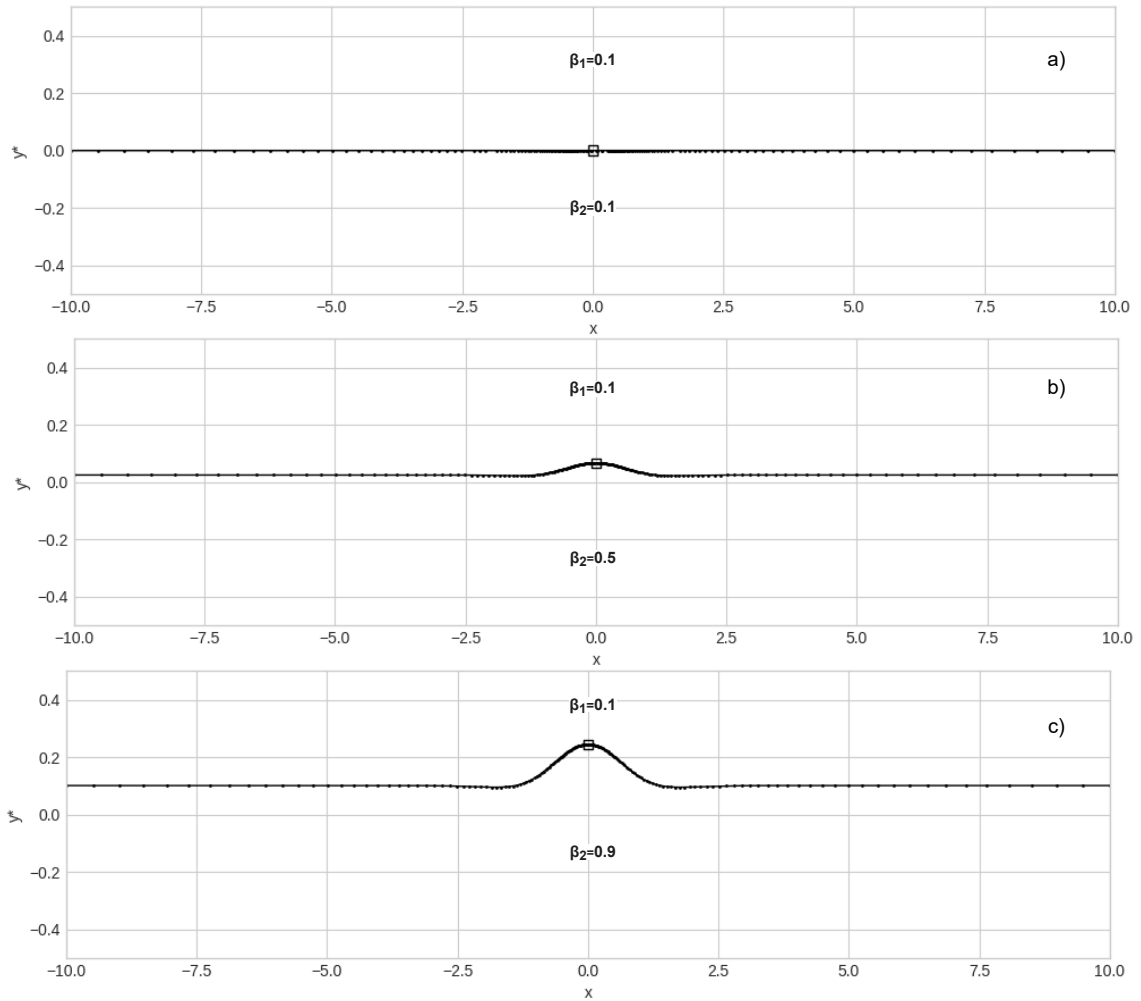


Figure 8.1: Interface representation for three flows with $\beta_1 = 0.1$, $De_1 = 0.33$, $De_2 = 0.33$, $Re=0.01$ and $Ca=\infty$. a) $\beta_2 = 0.1$; b) $\beta_2 = 0.5$; c) $\beta_2 = 0.9$

A better representation of the relation between the β ratios, Wi_0 and stagnation point displacement can be seen in Figure 8.2, where is shown that a higher stagnation point displacement is related to the β_1/β_2 ratio increase. The local Weissenberg number generally increases for higher β_1 values as showcased in the unitary ratio region and a higher local Weissenberg number concentration is found where the β ratio distances itself from an unitary value.

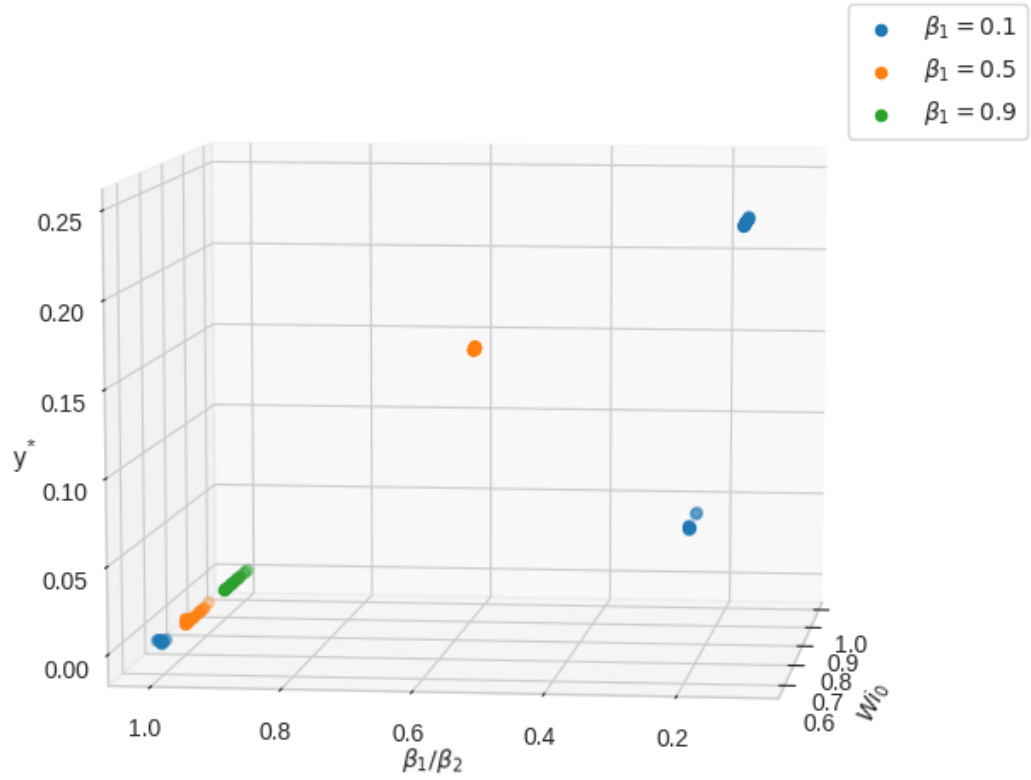


Figure 8.2: Plotting of the local Weissenberg, β ratio and stagnation point displacement relation for the flows with $De_1 = 0.29$, $Re=0.01$ and $Ca=\infty$. H stands for channel width ; $Dist$ stands for stagnation point distance from the geometry center.

The colours presented in this plot reveal similar local Weissenberg numbers, however all the represented values are present in Table 8.1. The plots regarding other De_1 values are practically identical as one can observe in Tables 8.2 to 8.5 making their graphical representation obsolete.

Chapter 9

Conclusions

The present study was initiated with the purpose of evaluating purely elastic phenomena of viscoelastic fluid flows in a cross-slot micro-geometry, utilizing its common setting of sharp edges and long inlet and outlet channels in order to achieve fully developed flows in each one of these channels.

The main objective was to analyse non-homogeneous concentration effects in flows with increasingly higher elasticity, with the purpose of understanding the physical motives that cause the two types of behavioural flow transitions previously reported as well as to identify the critical conditions in which they occur. A numerical analysis with the UCM constitutive model and a Newtonian fluid was made. After that the fully UCM model flow was investigated, in this case with different Deborah numbers in each inlet as a means to add to the single-phase results found in the literature [6]. And finally the effects of the addition of a viscous solvent in these flows were analysed for the Oldroyd-B constitutive model flow, also taking different fluid characteristics between channels into consideration.

In the Newtonian and UCM model type flow the inertialess and low Reynolds number flows were compared and present results with general good agreement. Both these cases and in the unitary Reynolds number flow, only one critical transition was found, namely from a steady symmetric state to an unsteady state. In this flow setting the Capillary number effects were also analysed, where the study of four near critical Deborah number flows suggest that the introduction of surface tension can have both a stabilizing and a destabilizing effect near critical conditions. The stabilizing effect is promoted by a low Capillary number or a high surface tension verified in all four case studies, meaning that high surface tensions could be considered for this purpose. For the $De=0.45$ flow an unsteady region is found between the Capillary values of 0.125 and 0.50. Although more research around these flow conditions is suggested, a potential explanation for the limited unsteady region could be the existence of a range of surface tension values that are high enough to promote unsteadiness (as this phenomenon is stress related) but not high enough to stabilize the interface of the flow phases. This is a mere suggestion supported by the fact that this region only exists for the near critical $De=0.45$. The final analysis of flows involving these two types of fluid was the

Reynolds ratio influence ($Reynolds\ ratio = \frac{Re_{UCM}}{Re_N}$), where one of the inlet flows was set to have an unitary Reynolds number, varying the other. In both cases the unsteady flow transition tends to occur at higher Deborah numbers when the Reynolds ratio is closer to an unitary value. For the Newtonian flow with an unitary Reynolds number, high Reynolds number ratios promote unsteadiness as it is reached at lower Deborah numbers and even though the same happens when this ratio is low, the flow is considerably more stable indicating that the low inertial effects contribute to this end. For the UCM model flow with an unitary Reynolds number the steady state limits of a given ratio are very close to its inverse ratio value, suggesting that the low inertial effects of the Newtonian fluid, do not contribute towards flow steadiness, further complementing the idea that these transitions are elastically driven. Another interesting finding was the presence of a steady asymmetric flow region for a Reynolds ratio $= 10^{-2}$ for the fixed $Re_{UCM} = 1$, that was afterwards attempted to map but with no success encouraging future attempts. A final observed tendency in this type of flow was the absence of a central stagnation point as the Newtonian side of the interface tends to swell in the symmetric steady cases, as well as a general local Weissenberg increase with the Deborah number increase.

In the multiphase UCM model fluid flow, a low Reynolds number with surface tension absence was considered, having the effects of different Deborah numbers in each inlet explored. The obtained results show good correspondence with the benchmark single phase cases [6], as five near critical Deborah numbers were fixed for one fluid and incrementally altered for the other, creating a flow behaviour map. In terms of stagnation point position it was verified that this drifted from the central point of the geometry as the Deborah number difference between the two fluids tends to increase. It is believed that this was the cause of a different tendency regarding local Weissenberg values, as they accompany the stagnation point displacement proclivity. This increase led to a completely asymmetric flow development, unlike near steady symmetric flows, where one outlet channel interface is the inverse function of the other. The interface state of near steady symmetric flows was also mapped, revealing an increment in asymmetry as the Deborah number increases also supported by the respective velocity profiles. Finally a stress field demonstration and a stress evaluation in the stagnation point is made, supporting that asymmetric state transition imposes a polymeric extra-stress relief. A stress/relaxation time correlation is also presented however it is highly suggested for future work that stress studies according to the present flow setting are made in a way that can quantify properties of the whole field of interest.

For the multiphase Oldroyd-B model flow the solvent viscosity effect was evaluated for the β values of 0.1, 0.5 and 0.9, plus the UCM/Oldroyd-B flow ($\beta_{UCM} = 0$). Unsteady flows and steady asymmetric flows were only reported for the $\beta_1 = 0$ cases, whereas for the Oldroyd-B/Oldroyd-B simulated flows, the flow was always steady and symmetric revealing a stabilizing nature of the added solvent. The Deborah number flow influence was also reported as very little, in terms of flow steadiness, local Weissenberg values and stagnation point displacement, meaning that the relaxation time influence is less noticeable for the studied cases. A distancing of the stagnation point position from the center of the geometry was also observed as the β_1/β_2 ratio decreases being very noticeable in the interface of these flows. Finally a higher local Weissenberg number

concentration is reported for flows with lower β_1/β_2 ratios, meaning that for these cases the Deborah number influence is lower. Due to time limitations the evaluated Deborah number range of the Oldroyd-B flows was not high enough to characterize the unsteady multiphase Oldroyd-B flows, which is highly suggested by the author.

All of the simulations performed in this thesis have their local Weissenberg numbers exposed so that they can be used for benchmarking purposes on this level of mesh refinement. For future work, it is also suggested that similar studies are conducted for different fluids, as the combinations *RheoTool* allows are immense and can allow a more complete comprehension on the nature of instabilities in viscoelastic fluid flows, for 2D and 3D structures.

Appendix A

RheoInterFoam Files

A.1 blockMeshDict

```

=====
|  \ \  /  /  F i e l d      | OpenFOAM: The Open Source CFD Toolbox      |
|  \ \  /  /  O peration    | Version: 7.0                               |
|  \ \  /  /  A nd          | Website: https://openfoam.org      |
|  \ \  /  /  M anipulation  |                                         |
|-----|
/*-----*/
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       blockMeshDict;
}
// *****

convertToMeters 1;

vertices
(
    (-10 -0.5 0)
    (-0.5 -0.5 0)
    (-0.5 -10 0)
    (0.5 -10 0)
    (0.5 -0.5 0)
    (10 -0.5 0)
    (10 0.5 0)
    (0.5 0.5 0)
    (0.5 10 0)
    (-0.5 10 0)
    (-0.5 0.5 0)
    (-10 0.5 0) // 11

    (-10 -0.5 1) //12
    (-0.5 -0.5 1)
    (-0.5 -10 1)
    (0.5 -10 1)
    (0.5 -0.5 1)
    (10 -0.5 1)
    (10 0.5 1)
    (0.5 0.5 1)
    (0.5 10 1)
    (-0.5 10 1)
    (-0.5 0.5 1)
    (-10 0.5 1) // 21

);

```

```

blocks
(
    hex (0 1 10 11 12 13 22 23) (60 51 1) simpleGrading (0.04 1 1) //1
    hex (2 3 4 1 14 15 16 13) (51 60 1) simpleGrading (1 0.04 1) //2
    hex (4 5 6 7 16 17 18 19) (60 51 1) simpleGrading (25 1 1) //3
    hex (10 7 8 9 22 19 20 21) (51 60 1) simpleGrading (1 25 1) //4
    hex (1 4 7 10 13 16 19 22) (51 51 1) simpleGrading (1 1 1) //8

);

edges
(
);

boundary
(
    inlet_north
    {
        type patch;
        faces
        (
            (9 8 20 21)
        );
    }

    inlet_south
    {
        type patch;
        faces
        (
            (2 3 15 14)
        );
    }

    outlet_east
    {
        type patch;
        faces
        (
            (5 6 18 17)
        );
    }
);

```

```

outlet_west
{
    type patch;
    faces
    (
        (0 11 23 12)
    );
}

walls
{
    type wall;
    faces
    (
        (0 1 13 12)
        (2 1 13 14)
        (3 4 16 15)
        (4 5 17 16)
        (7 6 18 19)
        (7 8 20 19)
        (10 9 21 22)
        (11 10 22 23)
    );
}

frontAndBack
{
    type empty;
    faces
    (
        (0 1 10 11)
        (2 3 4 1)
        (4 5 6 7)
        (10 7 8 9)
        (1 4 7 10)

        (12 13 22 23)
        (14 15 16 13)
        (16 17 18 19)
        (22 19 20 21)
        (13 16 19 22)
    );
}

);

mergePatchPairs
(
);

```

A.2 boundary

```

/*----- C++ -----*\
=====
\\  / F ield      | OpenFOAM: The Open Source CFD Toolbox
\\  / O peration  | Website: https://openfoam.org
\\  / A nd        | Version: 7
\\  / M anipulation |
\*-----*/
FoamFile
{
    version      2.0;
    format       ascii;
    class        polyBoundaryMesh;
    location     "constant/polyMesh";
    object       boundary;
}
// *****

6
(
    inlet_north
    {
        type            patch;
        nFaces          51;
        startFace       29340;
    }
    inlet_south
    {
        type            patch;
        nFaces          51;
        startFace       29391;
    }
    outlet_east
    {
        type            patch;
        nFaces          51;
        startFace       29442;
    }

    outlet_west
    {
        type            patch;
        nFaces          51;
        startFace       29493;
    }
    walls
    {
        type            wall;
        inGroups        List<word> 1(wall);
        nFaces          480;
        startFace       29544;
    }
    frontAndBack
    {
        type            empty;
        inGroups        List<word> 1(empty);
        nFaces          29682;
        startFace       30024;
    }
)

```

A.3 constitutiveProperties

```

/*-----* C++ *-----*/
|=====|
| \ \ / / | F i e l d | OpenFOAM: The Open Source CFD Toolbox |
| \ \ / / | O p e r a t i o n | Version: 7.0 |
| \ \ / / | A n d | Website: https://openfoam.org |
| \ \ / / | M a n i p u l a t i o n | |
|=====|
/*-----*/
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       constitutiveProperties;
}
// *****

phases (water air);

water
{
    parameters
    {
        type      Oldroyd-BLog;
        rho       rho [ 1 -3 0 0 0 0 0 ] 0.01;
        etaS       etaS [1 -1 -1 0 0 0 0] 0;
        etaP       etaP [1 -1 -1 0 0 0 0] 1;
        lambda     lambda [0 0 1 0 0 0 0] 0.40;

        stabilization coupling;
    }
}

air
{
    parameters
    {
        type      Newtonian;
        rho       rho [1 -3 0 0 0 0 0] 0.01;
        eta       eta [1 -1 -1 0 0 0 0] 1;
    }
}

sigma      sigma [ 1 0 -2 0 0 0 0 ] 1;

// *****

```

A.4 controlDict

```

=====
|  \ \  /  F ield      | OpenFOAM: The Open Source CFD Toolbox |
|  \ \  /  O peration  | Version: 7.0                        |
|  \ \  /  A nd        | Website: https://openfoam.org |
|  \ \  /  M anipulation|                                     |
|*-----*|
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    location     "system";
    object       controlDict;
}
// *****

application      rheoInterFoam;

startFrom        latestTime;

startTime        0;

stopAt           endTime;

endTime          120;

deltaT           0.00001;

writeControl     adjustableRunTime;

writeInterval    10;

purgeWrite       0;

writeFormat      ascii;

writePrecision   6;

writeCompression uncompressed;

timeFormat       general;

timePrecision    6;

runTimeModifiable yes;

adjustTimeStep   yes;

maxCo            0.1;

maxAlphaCo       1;

maxDeltaT        1;

```

```

functions
{
    outputCd
    {
        functionObjectLibs ("libutilityFunctionObjects.so");
        type coded;
        redirectType calCd;
        writeControl timeStep;
        writeInterval 100;
        enabled yes;

        codeWrite
        #{
            // Lookup/create variable

            const volVectorField& U = mesh().lookupObject<volVectorField>("U");

            scalarList list;

            label cellC = mesh().findCell(vector(0,0,0.5));

            // Compute Ux

            list.append(mesh().time().value()); // Time (col 0)
            list.append(U[cellC].x() / (Foam::sqrt( Foam::max(1e-12, U[cellC].x()*U[cellC].x()-
+U[cellC].y()*U[cellC].y() ) ) ));// Ux/Umag (col 1)

            // Write data

            string comsh;
            string filename("U.txt");
            std::stringstream doub2str; doub2str.precision(12);

            comsh = "./writeData " + filename;
            forAll(list, id)
            {
                doub2str.str(std::string());
                doub2str << list[id];
                comsh += " " + doub2str.str();
            }
        }
    }
}

```

```
    if (Pstream::master())
    {
        system(comsh);
    }

    if (Pstream::parRun())
    {
        for (label procI = 0; procI < Pstream::nProcs(); procI++)
        {
            if (procI == Pstream::myProcNo())
            {
                if (cellC!=-1)
                {
                    system(comsh);
                }
            }
        }
    }
    else
    {
        system(comsh);
    }

    #};
}

}
```

A.5 fvSchemes

```

=====
| \ \ / / | F i e l d | OpenFOAM: The Open Source CFD Toolbox |
| \ \ / / | O p e r a t i o n | Version: 7.0 |
| \ \ / / | A n d | Website: https://openfoam.org |
| \ \ / / | M a n i p u l a t i o n |
|*-----*/
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       fvSchemes;
}
// *****

ddtSchemes
{
    default      Euler;
}

gradSchemes
{
    default      Gauss linear;
    grad(p)      Gauss linear;
    grad(U)      Gauss linear;
    linExtrapGrad Gauss linear;
}

divSchemes
{
    default      none;
    div(Sum(tau)) Gauss linear;
    div(grad(U))  Gauss linear;

    div(rhoPhi,U) Gauss limitedLinearV 1;

    div(phi,alpha) Gauss vanLeer;
    div(phiIrb,alpha) Gauss linear;

    div(eta*alpha*dev2(T(gradU))) Gauss linear;

    div(phi,theta.water) GaussDefCmpw cubista;
    div(phi,theta.air) GaussDefCmpw cubista;
    div(phi,tau.water) GaussDefCmpw cubista;
    div(phi,tau.air) GaussDefCmpw cubista;
}

```

```
laplacianSchemes
{
    default                Gauss linear corrected;
}

interpolationSchemes
{
    default                linear;
}

snGradSchemes
{
    default                corrected;
}

fluxRequired
{
    default                no;
    p_rgh;
    pcorr;
    alpha1;
}

// ***** //
```

A.6 fvSolution

```

=====
|  \ \  /  | F ield      | OpenFOAM: The Open Source CFD Toolbox      |
|  \ \  /  | O peration  | Version: 7.0                          |
|  \ \  /  | A nd        | Website: https://openfoam.org      |
|  \ \  /  | M anipulation|                                     |
|*-----*|
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       fvSolution;
}
// *****

solvers
{
    "alpha.water.*"
    {
        nAlphaCorr      2;
        nAlphaSubCycles 2;
        cAlpha          1;

        MULESCorr       yes;
        nLimiterIter     3;

        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-8;
        relTol          0;
    }

    "pCorr.*"
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-10;
        relTol          0;
    }

    p_rgh
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-10;
        relTol          0.0;
    }
}

```

```

p_rghFinal
{
    $p_rgh;
    relTol          0;
}

"(theta.*|tau.*|U)"
{
    solver          PBiCG;
    preconditioner
    {
        preconditioner DILU;
    }

    tolerance        1e-10;
    relTol           0;
    minIter          0;
    maxIter          1000;
}
}

PIMPLE
{
    nInIter          1;
    SIMPLEC          true;

    nCorrectors      1;
    nNonOrthogonalCorrectors 0;
}

relaxationFactors
{
    equations
    {
        ".*" 1;
    }
}

```

A.7 decomposeParDict

```

=====
|  \ \  /  /  F ield      | OpenFOAM: The Open Source CFD Toolbox |
|  \ \  /  /  O peration  | Version: 7.0                        |
|  \ \  /  /  A nd        | Website: https://openfoam.org |
|  \ \  /  /  M anipulation|                                     |
\*------*/
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       decomposeParDict;
}
// *****

numberOfSubdomains 8;

method          scotch;

// *****

```

A.8 setFields

```

/*-----*-----*-----*-----*-----*-----*-----*-----*-----*
|=====| F i e l d | O p e n F O A M : T h e O p e n S o u r c e C F D T o o l b o x |
| \ \ / | O p e r a t i o n | V e r s i o n : 7 . 0 |
| \ \ / | A n d | W e b s i t e : https://openfoam.org |
| \ \ / | M a n i p u l a t i o n | |
|=====| |
\*-----*-----*-----*-----*-----*-----*-----*-----*-----*

FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       setFieldsDict;
}
// *****

defaultFieldValues
(
    volScalarFieldValue alpha.water 0
);

regions
(
    boxToCell
    {
        box (-20 0 -1) (20 20 1);
        fieldValues
        (
            volScalarFieldValue alpha.water 1
        );
    }
);

// *****

```

Appendix B

Post-processing code

```
1 #include <stdio.h>
2 #include <math.h>
3 #include <stdlib.h>
4 #include <string.h>
5
6 #define BUFFER_SIZE 1000
7
8 //Function Declaration
9 float find_lambda_n(FILE *cp, const char *word, float lambda_n);
10 float find_lambda_s(FILE *cp, const char *word, float lambda_s);
11
12 int main() {
13 //Used variables
14 FILE *fp, *f, *f_read, *y_asterisco, *cp, *cc, *tens_x0, *tens_parede, *tens_int, *inter_est, *perfisV;
15 //Vars_Points_Filter
16 int i,j,k2,k_cima,l=1,k1=0,comma=1,inter_count=0,lines=0,cellCenters_check=1,lambda_check=1;
17 long double x1,y1,z1,x2,y2,alpha1,alpha2,alpha_cima,x_cima,y_cima,inter,alpha_ant=10,alpha_int;
18 char c,ch;
19 //Vars_Position_Indicators
20 int alpha_ind=0,x_ind=0,tau_wateryy_ind=0,tau_waterxx_ind,tauMFxx_ind=0,tauMFyy_ind=0,count_inter=1;
21 char read[30];
22 //Vars_Find_Lambda
23 char word[7]="lambda";
24 float lambda_n,lambda_s,lambda;
25 int numLambdas;
26 //Vars_Find_dq
27 long double dq=0,x_aux,inter_aux,x_aux2,inter_aux2,q1,q2;
28 //Interface point in x=0
29 long double x_0,y_x0,x_0_ant=-1000,x_0_post=1000,y_x0_ant,y_x0_post;
30 int x_0_check=0;
31 //Vars_Find_Wl0
32 long double u,v,w,x_wi,y_wi,wi,dudx,dudy,dvdx;
33 long double x_n3,y_n3=1000,u_n3,v_n3,x_s3,y_s3=-1000,u_s3,v_s3,x_e3=1000,y_e3,u_e3,v_e3,x_o3=-1000,y_o3,u_o3,v_o3,d_ne,d_no,d_se,
d_so,x3,y3;
```

```

34 //Stagnation Point
35 long double x_est,y_est,u1,v1,u2,v2,u_cima,v_cima,u_inter,v_inter;
36 int pe=-1;
37 //Error condition in the stagnation point
38 int notThere=0;
39 //Vars_Find_stresses
40 long double tau_wateryy, tau_waterxx, tauMFyy, tauMFxx;
41 //Vars_Find_Stagnation_Point
42 long double
u_min_mag,v_min_mag,x_min_mag,inter_min_mag,u_dir_mag,v_dir_mag,x_dir_mag=1000,inter_dir_mag,u_esq_mag,v_esq_mag,x_esq_mag=-1000,
43 inter_esq_mag,v_min_mag=1000,vel_mag,x_P,inter_P,u_P,v_P,u_est,v_est,x_est_por_v;
44 //Vars_refined_case
45 int refineCheck=0,fileSize;
46 //General point filtering
47 fp=fopen("pontos0.txt", "r");
48 f=fopen("f.txt", "w");
49 perfisV=fopen("perfis.txt", "w");
50 //Non-opening of the file condition
51 if (fp==NULL)
52 {
53     printf("Could not open file: pontos.txt\n");
54     return 0;
55 }
56 //Finding used variable positions in the first line of the program
57 for (ch=getc(fp); ch != EOF ; ch=getc(fp))
58 {
59     if(ch=='\n')
60     {
61         break;
62     }
63     else if(ch == ',')
64     {

```

```

65 //Comparing read string
66 if(strstr(read,"alpha.water"))
67 {
68     alpha_ind=comma;
69 }
70 if(strstr(read,"tau.water:0"))
71 {
72     tau_waterxx_ind=comma;
73 }
74 if(strstr(read,"tau.water:1"))
75 {
76     tau_wateryy_ind=comma;
77 }
78 if(strstr(read,"tauMF:0"))
79 {
80     tauMFxx_ind=comma;
81 }
82 if(strstr(read,"tauMF:1"))
83 {
84     tauMFyy_ind=comma;
85 }
86 if(strstr(read,"Points:0"))
87 {
88     x_ind=comma;
89     break;
90 }
91 //Cleaning the string
92 read[0] = '\0';
93 comma++;
94 }
95 else
96 {
97     strncat(read,&ch,1);
98 }
99

```

```

100 if((x_ind==0) || (alpha_ind==0) || (tauMfxx_ind==0) || (tauMfyy_ind==0) || (tau_waterxx_ind==0) || (tau_wateryy_ind==0))
101 {
102     printf("Problem with indicators\n");
103     return 0;
104 }
105 rewind(fp);
106 lines=0;
107 comma=0;
108 //General point filtering cycle
109 for (ch=getc(fp); ch != EOF; ch=getc(fp))
110 {
111     if(ch=='\n')
112     {
113         comma=2;
114         lines++;
115         if(lines!=0)
116             fscanf(fp,"%Lf,%Lf", &u, &v);
117     }
118     else if(ch == ',')
119     {
120         comma++;
121         if((comma==alpha_ind) && (lines!=0))
122         {
123             fscanf(fp, "%Lf", &alpha1);
124         }
125         if((comma==x_ind) && (lines!=0))
126         {
127             fscanf(fp, "%Lf,%Lf,%Lf", &x1, &y1, &z1);
128             if(((z1==0) && (y1<0.5) && (y1>-0.5) && (alpha1>=0.0001) && (alpha1<=0.9999) && (x1<=3) && (x1 >=-3) && (alpha1!=
=alpha_ant)))
129             || ((z1==0) && (y1<0.5) && (y1>-0.5) && (alpha1>=0.0001) && (alpha1<=0.9999) && (x1>3))
130             || ((z1==0) && (y1<0.5) && (y1>-0.5) && (alpha1>=0.0001) && (alpha1<=0.9999) && (x1<-3)))
131             {
132                 alpha_ant=alpha1;
133                 fprintf(f,"%Lf %Lf %Lf %Lf %Lf",alpha1,x1,y1,u,v);

```



```

134                 fprintf(f, " %d\n",l);
135                 l++;
136             }
137             if(((x1==0.5)||((x1==0.5)&&((y1<=0.5)&&(y1>=-0.5))&&(z1==0)))
138             {
139                 fprintf(perfisv,"%Lf %Lf %Lf\n",x1,y1,u);
140             }
141         }
142     }
143 }
144 fclose(f);
145 fileSize=lines;
146
147 //Interface point filtering
148 f_read=fopen("f.txt", "r");
149 y_asterisco=fopen("y*.txt", "w");
150
151 for(i=1;i<l;i++)
152 {
153     if(i==1)
154         fscanf(f_read,"%Lf %Lf %Lf %Lf %Lf %d\n", &alpha1, &x1, &y1, &u1, &v1, &k1);
155
156     while(k1!=i) //Finds every point: "1"
157     {
158         fscanf(f_read,"%Lf %Lf %Lf %Lf %Lf %d\n", &alpha1, &x1, &y1, &u1, &v1, &k1);
159     }
160
161     rewind(f_read);
162     y_cima=1000;
163     k_cima=0;
164     for(j=1;j<l;j++)
165     {
166         fscanf(f_read,"%Lf %Lf %Lf %Lf %Lf %d\n", &alpha2, &x2, &y2, &u2, &v2, &k2);
167         //Searches the above neighbour of each point "1": "2"
168         if((x1==x2)&&(y2>y1)&&((y_cima-y1)>(y2-y1)))

```

```

169         {
170             alpha_cima=alpha2;
171             x_cima=x2;
172             y_cima=y2;
173             u_cima=u2;
174             v_cima=v2;
175             k_cima=j;
176         }
177     }
178     if(k_cima!=0)
179     {
180         if(((alpha1-0.5)*(alpha_cima-0.5)<0)&&(alpha_cima>alpha1))
181         {
182             inter=y1+(0.5-alpha1)*(y_cima-y1)/(alpha_cima-alpha1);
183             inter_count++;
184             u_inter=(u1*(inter-y1)+u_cima*(y_cima-inter))/(y_cima-y1);
185             v_inter=(v1*(inter-y1)+v_cima*(y_cima-inter))/(y_cima-y1);
186             alpha_int=alpha_cima-alpha1;
187             fprintf(y_asterisco,"%Lf %Lf %Lf %Lf %Lf\n", u_inter, v_inter, alpha_int, x1, inter);
188             count_inter++;
189             //Finding the flow asymmetry parameter points
190             if(x1==10)
191             {
192                 x_aux=x1;
193                 inter_aux=inter;
194             }
195             if(x1==-10)
196             {
197                 x_aux2=x1;
198                 inter_aux2=inter;
199             }
200             //Interface point in x=0
201             if(x1==0)
202             {
203                 x_0=x1;

```

```

204                 y_x0=inter;
205                 x_0_check=1;
206             }
207             else if((x1<0) && (x1>x_0_ant))
208             {
209                 x_0_ant=x1;
210                 y_x0_ant=inter;
211             }
212             else if((x1>0) && (x1<x_0_post))
213             {
214                 x_0_post=x1;
215                 y_x0_post=inter;
216             }
217         }
218     }
219     rewind(f_read);
220 }
221 fclose(y_asterisco);
222 fclose(f_read);
223 if(fileSize>50000)
224     refineCheck=1;
225 if(refineCheck==0)//No refinement code
226 {
227     //Stagnation point finding
228     inter_est=fopen("y*.txt", "r");
229     //Lowest velocity magnitude point
230     for(j=1;j<count_inter;j++)
231     {
232         fscanf(inter_est,"%Lf %Lf %Lf %Lf %Lf\n", &u_inter, &v_inter, &alpha_int, &x1, &inter);
233         vel_mag=u_inter*u_inter+v_inter*v_inter;
234         if(vel_mag<vel_min_mag)
235         {
236             u_min_mag=u_inter;
237             v_min_mag=v_inter;
238             x_min_mag=x1;

```

```

239         inter_min_mag=inter;
240         vel_min_mag=vel_mag;
241     }
242 }
243 rewind(inter_est);
244 //Neighbour points to the lowest velocity magnitude point
245 for(j=1;j<count_inter;j++)
246 {
247     fscanf(inter_est,"%Lf %Lf %Lf %Lf %Lf\n", &u_inter, &v_inter, &alpha_int, &x1, &inter);
248     if((x1>x_min_mag) && (x1<x_dir_mag))
249     { //Right
250         x_dir_mag=x1;
251         inter_dir_mag=inter;
252         u_dir_mag=u_inter;
253         v_dir_mag=v_inter;
254     }
255     if((x1<x_min_mag) && (x1>x_esq_mag))
256     { //Left
257         x_esq_mag=x1;
258         inter_esq_mag=inter;
259         u_esq_mag=u_inter;
260         v_esq_mag=v_inter;
261     }
262 }
263 //Finding in between each of these two and the lowest velocity magnitude point, is the stagnation point called P
264 if(fabs(u_min_mag)<0.000001)
265 { //Condition of already having the stagnation point
266     x_est=x_min_mag;
267     y_est=inter_min_mag;
268     u_est=u_min_mag;
269     v_est=v_min_mag;
270     printf("O ponto de menor magnitude É o ponto de estagnação!\n");
271 }
272 else
273
274     if((u_min_mag*u_dir_mag<0) && (v_min_mag*v_dir_mag<0))
275     { //Stagnation point to the right of the lowest velocity magnitude point
276         x_P=x_dir_mag;
277         inter_P=inter_dir_mag;
278         u_P=u_dir_mag;
279         v_P=v_dir_mag;
280         pe=1;
281     }
282     else if((u_min_mag*u_esq_mag<0) && (v_min_mag*v_esq_mag<0))
283     { //Stagnation point to the left of the lowest velocity magnitude point
284         x_P=x_esq_mag;
285         inter_P=inter_esq_mag;
286         u_P=u_esq_mag;
287         v_P=v_esq_mag;
288         pe=0;
289     }
290     else
291     {
292         pe=2;
293         notThere=1;
294     }
295
296     if(notThere==0)
297     {
298         x_est=(u_min_mag*x_P-u_P*x_min_mag)/(u_min_mag-u_P);
299         y_est=inter_min_mag-((inter_min_mag-inter_P)*(x_min_mag-x_est))/(x_min_mag-x_P);
300         //((u_min_mag*inter_P-u_P*inter_min_mag)/(u_min_mag-u_P); same thing
301         x_est_por_v=(v_min_mag*x_P-v_P*x_min_mag)/(v_min_mag-v_P);
302     }
303 }
304 if(notThere==1) //Condition for the stagnation point not being in the vicinity of the interface point with lowest velocity magnitude
305 {
306     rewind(inter_est);
307     if(u_min_mag<0) //Negative velocity, find stagnation point in the right of the lowest velocity magnitude point
308     {

```

```

309     u_esq_mag=u_min_mag;
310     v_esq_mag=v_min_mag;
311     x_esq_mag=x_min_mag;
312     inter_esq_mag=inter_min_mag;
313     u_dir_mag=-1;
314     while(u_dir_mag*u_esq_mag>0)
315     {
316         x_dir_mag=1000;
317         for(j=1;j<count_inter;j++)
318             { //Finding point to the right
319                 fscanff(inter_est,"%Lf %Lf %Lf %Lf %Lf\n", &u_inter, &v_inter, &alpha_int, &x1, &inter);
320                 if(x1>x_esq_mag)
321                 {
322                     if(x1<x_dir_mag)
323                     {
324                         x_dir_mag=x1;
325                         inter_dir_mag=inter;
326                         u_dir_mag=u_inter;
327                         v_dir_mag=v_inter;
328                     }
329                 }
330             }
331         if(u_dir_mag*u_esq_mag>0)
332             { //Update of the left point if the right point does not have a 'u' of different signal
333                 u_esq_mag=u_dir_mag;
334                 v_esq_mag=v_dir_mag;
335                 x_esq_mag=x_dir_mag;
336                 inter_esq_mag=inter_dir_mag;
337             }
338     }
339 }
340 else if(u_min_mag>0) //Negative velocity, find stagnation point in the left of the lowest velocity magnitude point
341 {
342     u_dir_mag=u_min_mag;
343     v_dir_mag=v_min_mag;

```

```

344     x_dir_mag=x_min_mag;
345     inter_dir_mag=inter_min_mag;
346     u_esq_mag=1;
347     while(u_dir_mag*u_esq_mag>0)
348     {
349         x_esq_mag=-1000;
350         for(j=1;j<count_inter;j++)
351             { //Finding point to the left
352                 fscanff(inter_est,"%Lf %Lf %Lf %Lf %Lf\n", &u_inter, &v_inter, &alpha_int, &x1, &inter);
353                 if(x1<x_dir_mag)
354                 {
355                     if(x1>x_esq_mag)
356                     {
357                         x_esq_mag=x1;
358                         inter_esq_mag=inter;
359                         u_esq_mag=u_inter;
360                         v_dir_mag=v_inter;
361                     }
362                 }
363             }
364         if(u_dir_mag*u_esq_mag>0) //Update of the right point if the left point does not have a 'u' of different signal
365         {
366             u_dir_mag=u_esq_mag;
367             v_dir_mag=v_esq_mag;
368             x_dir_mag=x_esq_mag;
369             inter_dir_mag=inter_esq_mag;
370         }
371     }
372 }
373 x_est=(u_dir_mag*x_esq_mag-u_esq_mag*x_dir_mag)/(u_dir_mag-u_esq_mag);
374 y_est=(u_dir_mag*inter_esq_mag-u_esq_mag*inter_dir_mag)/(u_dir_mag-u_esq_mag);
375 if(fabs(x_est)<0.0001)
376     x_est=0;
377 if(fabs(y_est)<0.0001)
378     y_est=0;

```

```

379     }
380     printf("\n--Stagnation point--\n");
381     if(pe==1) //Error condition
382         printf("ERROR FINDING THE STAGNATION POINT\n");
383     printf("Minimal velocity magnitude interface point: (x,y)=(%Lf,%Lf) ;; (u,v)=(%Lf,%Lf)
\n",x_min_mag,inter_min_mag,u_min_mag,v_min_mag);
384     printf("Interface point to its right: (x,y)=(%Lf,%Lf) ;; (u,v)=(%Lf,%Lf)\n",x_dir_mag,inter_dir_mag,u_dir_mag,v_dir_mag);
385     printf("Interface point to its left: (x,y)=(%Lf,%Lf) ;; (u,v)=(%Lf,%Lf)\n",x_esq_mag,inter_esq_mag,u_esq_mag,v_esq_mag);
386     if(pe==2)
387         printf("The stagnation point is not in the vicinity of the minimal velocity magnitude point\n");
388     if(pe==1)
389         printf("The stagnation point is located to the right of the minimal velocity magnitude point\n");
390     if(pe==0)
391         printf("The stagnation point is located to the left of the minimal velocity magnitude point\n");
392     printf("\nStagnation point: (x,y)=(%Lf,%Lf)\n\n",x_est,y_est);
393
394 //Interface Analysis
395     printf("--Interface analysis--\nThe number of interface points is: %d\nFor the Interface Representation run command:\npython3
graphv2.py &\n\n", inter_count);
396
397 //Flow asymmetry parameter
398     printf("--Flow asymmetry parameter--\n");
399     //THIS POINT IS NOT USED BUT CAN BE USEFUL
400     if(x_0_check!=1)
401     {
402         x_0=0;
403         y_x0=(y_x0_ant*fabsl(x_0_ant)+y_x0_post*fabsl(x_0_post))/(fabsl(x_0_ant)+fabsl(x_0_post));
404     }
405     if((inter_aux==0)|| (inter_aux2==0))
406         printf("There is no interface point for x=10 or for x=-10, check the problem\n\n");
407     else
408     {
409         printf("Relevant interface positions:\nLeft (x=%0.1Lf): y=%0.12Lf ;; Right (x=%0.1Lf):
y=%0.12Lf\n",x_aux2,inter_aux2,x_aux,inter_aux);
410         q1=0.5*fabsl(inter_aux-y_est);

```

```

411         q2=1-q1;
412         dq=fabsl(q2-q1);
413         printf("q1 = %Lf ;; q2 = %Lf\n",q1,q2);
414         printf("|DQ| = %0.12Lf\n",dq);
415     }
416     printf("For the velocity profiles in x=0.5 and x=0.5 run command:\npython3 perfis05.py &\n\n");
417
418 //Local Weissenberg number
419     printf("--Local Weissenberg Number--\n");
420     //-- 1) Lambda
421     cp=fopen("../constant/constitutiveProperties","r");
422     if (cp==NULL)
423     {
424         printf("Could not open constitutive properties, check file directory.\n\n");
425         return 0;
426     }
427     lambda_n=find_lambda_n(cp,word,lambda_n);
428     lambda_s=find_lambda_s(cp,word,lambda_s);
429     if((lambda_n==1) && (lambda_s==1))
430     {
431         printf("Lambda not found: There is no Weissenberg number in a fully newtonian crossing\n\n");
432         lambda_check=0;
433         numLambdas=0;
434     }
435     else if((lambda_n!=1) && (lambda_s==1))
436     {
437         printf("Just one of the fluids is viscoelastic\n");
438         printf("Lambda = %f\n",lambda_n);
439         numLambdas=1;
440     }
441     else if((lambda_n!=1) && (lambda_s!=1))
442     {
443         printf("Both fluids are viscoelastic\n");
444         printf("Northern inlet Lambda = %f\nSouthern inlet Lambda = %f\n",lambda_n,lambda_s);
445         numLambdas=2;

```

```

446     }
447     else
448         printf("Something wrong with find_lambda\n");
449     //--// 2) Stresses
450     tens_x0=fopen("tens_x0.txt", "w");
451     tens_parede=fopen("tens_parede.txt", "w"); //Stress files
452     tens_int=fopen("tens_int.txt", "w");
453     cc=fopen("cc0.txt", "r");
454     if (cc==NULL)
455     {
456         printf("The Cell Centers file 'cc.txt' is required.\n\n");
457     }
458     lines=0;
459     for (ch=getc(cc); ch != EOF; ch=getc(cc)) //Stress search in cc.txt
460     {
461         if(ch=='\n')
462         {
463             comma=1;
464             if(lines!=0)
465                 fscanf(cc,"%Lf,%Lf", &u, &v);
466             lines++;
467         }
468         else if(ch == ',')
469         {
470             comma++;
471             if((comma==tau_waterxx_ind-1) && (lines!=0))
472             {
473                 fscanf(cc,"%Lf,%Lf", &tau_waterxx, &tau_wateryy);
474                 comma=comma+2;
475             }
476             else if((comma==tauMFxx_ind) && (lines!=0))
477             {
478                 fscanf(cc,"%Lf,%Lf", &tauMFxx, &tauMFyy);
479                 comma=comma+2;
480

```

```

481     }
482     else if((comma==(x_ind)) && (lines!=0))
483     {
484         fscanf(cc, "%Lf,%Lf", &x_wi, &y_wi);
485
486         if((x_wi==0)&&(lines!=1)) //Stress writing in x=0
487         {
488             fprintf(tens_x0,"%Lf %Lf %Lf %Lf %Lf\n", y_wi, tauMFyy, tau_wateryy, tau_waterxx, tauMFxx);
489         }
490         if((fabs(x_wi-0.49)<0.001)&&(lines!=1)&&(x_wi>0)) //Stress writing in the wall
491         {
492             fprintf(tens_parede,"%Lf %Lf %Lf\n", y_wi, tauMFyy, tau_wateryy);
493         }
494         if((fabs(x_wi-0.255)<0.001)&&(lines!=1)&&(x_wi>0)) //Stress writing between x=0 and the wall
495         {
496             fprintf(tens_int,"%Lf %Lf %Lf\n", y_wi, tauMFyy, tau_wateryy);
497         }
498     }
499 }
500
501 // --// 4) Wi_0 -> utilizes 4 cell centers closest to the cell center closest to the stagnation point
502
503 //Finding cell center closest to the stagnation point
504 d_ne=(x_est-x_ne1)*(x_est-x_ne1)+(y_est-y_ne1)*(y_est-y_ne1);
505 d_no=(x_est-x_no1)*(x_est-x_no1)+(y_est-y_no1)*(y_est-y_no1);
506 d_so=(x_est-x_so1)*(x_est-x_so1)+(y_est-y_so1)*(y_est-y_so1);
507 d_se=(x_est-x_se1)*(x_est-x_se1)+(y_est-y_se1)*(y_est-y_se1);
508
509 if((d_ne<=d_no)&&(d_ne<=d_so)&&(d_ne<=d_se))
510 {
511     x3=x_ne1;
512     y3=y_ne1;
513 }
514 else if((d_no<=d_so)&&(d_no<=d_se))
515

```

```

516         x3=x_no1;
517         y3=y_no1;
518     }
519     else if(d_so<=d_se)
520     {
521         x3=x_so1;
522         y3=y_so1;
523     }
524     else
525     {
526         x3=x_se1;
527         y3=y_se1;
528     }
529     if((fabsl(x_est)<0.001) && (fabsl(y_est)<0.001))//Distances algorithm works if we're not in (0,0)
530     { //Case of stagnation point being in x=0 and y=0
531         x3=x_est;
532         y3=y_est;
533     }
534     else if((fabsl(x_est)<0.001) && (fabsl(y_est)>0.001))
535     { //Searching point in x=0
536         x3=x_est;
537         y3=1000;
538         rewind(cc);
539         for (ch=getc(cc); ch != EOF; ch=getc(cc))
540         {
541             if(ch=='\n')
542             {
543                 comma=1;
544                 lines++;
545             }
546             else if(ch == ',')
547             {
548                 comma++;
549                 if((comma==x_ind) && (lines!=0))
550                 {

```

```

551                     fscanf(cc,"%Lf,%Lf", &x_wi, &y_wi);
552                     if(fabsl(x_wi)<0.001)
553                     {
554                         if(fabsl(y_wi-y_est)<fabsl(y3-y_est))
555                             y3=y_wi;
556                     }
557                 }
558             }
559         }
560     }
561     else if((fabsl(x_est)>0.001) && (fabsl(y_est)<0.001))
562     { //Searching point in y=0
563         y3=y_est;
564         x3=1000;
565         rewind(cc);
566         for (ch=getc(cc); ch != EOF; ch=getc(cc))
567         {
568             if(ch=='\n')
569             {
570                 comma=1;
571                 lines++;
572             }
573             else if(ch == ',')
574             {
575                 comma++;
576                 if((comma==x_ind) && (lines!=0))
577                 {
578                     fscanf(cc,"%Lf,%Lf", &x_wi, &y_wi);
579                     if(fabsl(y_wi)<0.001)
580                     {
581                         if(fabsl(x_wi-x_est)<fabsl(x3-x_est))
582                             x3=x_wi;
583                     }
584                 }
585             }

```

```

586     }
587 }
588
589 //Finding 4 cell centers adjacent to x3,y3
590 lines=0;
591 rewind(cc);
592 for (ch=getc(cc); ch != EOF; ch=getc(cc))
593 {
594     if(ch=='\n')
595     {
596         comma=2;
597         lines++;
598         if(lines!=0)
599             fscanf(cc,"%Lf,%Lf",&u,&v);
600     }
601     else if(ch == ',')
602     {
603         comma++;
604         if((comma==alpha_ind) && (lines!=0))
605         {
606             fscanf(cc, "%Lf", &alpha1);
607         }
608         if((comma==x_ind) && (lines!=0))
609         {
610             fscanf(cc, "%Lf,%Lf", &x_wi, &y_wi);
611             if((fabs(x_wi-x3)<0.0001) && (y_wi>y3)) //Values for cell center above point 3
612             {
613                 if(y_wi<y_n3)
614                 {
615                     x_n3=x_wi;
616                     y_n3=y_wi;
617                     u_n3=u;
618                     v_n3=v;
619                 }
620             }

```

```

621         else if((fabs(x_wi-x3)<0.0001) && (y_wi<y3)) //Values for cell center below point 3
622         {
623             if(y_wi>y_s3)
624             {
625                 x_s3=x_wi;
626                 y_s3=y_wi;
627                 u_s3=u;
628                 v_s3=v;
629             }
630         }
631         else if((fabs(y_wi-y3)<0.0001) && (x_wi>x3)) //Values for cell center to the right of point 3
632         {
633             if(x_wi<x_e3)
634             {
635                 x_e3=x_wi;
636                 y_e3=y_wi;
637                 u_e3=u;
638                 v_e3=v;
639             }
640         }
641         else if((fabs(y_wi-y3)<0.0001) && (x_wi<x3)) //Values for cell center to the left of point 3
642         {
643             if(x_wi>x_o3)
644             {
645                 x_o3=x_wi;
646                 y_o3=y_wi;
647                 u_o3=u;
648                 v_o3=v;
649             }
650         }
651     }
652 }
653 }
654 dudx=(u_e3-u_o3)/(x_e3-x_o3);
655 dudy=(u_n3-u_s3)/(y_n3-y_s3);

```

```

656         dwdx=(v_e3-v_o3)/(x_e3-x_o3);
657
658         printf("\nLocal Weissenberg calculations:\n");
659
660         if(numLambdas==1)
661             lambda=lambda_n;
662         if(numLambdas==2)
663         {
664             lambda=(lambda_n*(y_est-y_s3)+lambda_s*(y_n3-y_est))/(y_n3-y_s3);
665             printf("Lambda = %f\n",lambda);
666         }
667
668         w1=lambda*sqrt1(dudx*dudx+dvdv*dvdv);
669         printf("w10 = %Lf\n",w1);
670         printf("Closest cc to the stagnation point coordinates: (x,y)=(%Lf,%Lf)\n",x3,y3);
671         printf("Cell center N from the closest cc to the stagnation point: (x,y)=(%Lf,%Lf) (u,v)=(%Lf,%Lf)\n",x_n3,y_n3,u_n3,v_n3);
672         printf("Cell center O from the closest cc to the stagnation point: (x,y)=(%Lf,%Lf) (u,v)=(%Lf,%Lf)\n",x_o3,y_o3,u_o3,v_o3);
673         printf("Cell center E from the closest cc to the stagnation point: (x,y)=(%Lf,%Lf) (u,v)=(%Lf,%Lf)\n",x_e3,y_e3,u_e3,v_e3);
674         printf("Cell center S from the closest cc to the stagnation point: (x,y)=(%Lf,%Lf) (u,v)=(%Lf,%Lf)\n",x_s3,y_s3,u_s3,v_s3);
675     }
676 }
677 //Stress Analysis
678 printf("--Tensions--\nFor the Tension Analysis in 3 axes run command:\npython3 tensoes.py &\nFor the first normal stress difference along\nx=0 run command:\npython3 fnsd.py &\n\n");
679
680 //Closure of all open files
681 fclose(tens_x0);
682 fclose(tens_parde);
683 fclose(tens_int);
684 fclose(fp);
685 fclose(inter_est);
686 //After closing inter_est, y* is reopened to write the stagnation point position
687 inter_est=fopen("y*.txt","a");
688 fprintf(inter_est, "%Lf %Lf 0 0 0\n",x_est,y_est);
689 fclose(inter_est);

```

```

690 fclose(cc);
691 fclose(cp);
692 fclose(perfisV);
693 //Error analysis
694 //printf("Index of x: %d\nIndex of alpha: %d\nIndex of tau.wateryy: %d\nIndex of tau.waterxx: %d\nIndex of tauMFxx: %d\nIndex of tauMFyy: %d\n",x_ind,alpha_ind,tau_water_ind,tau_waterxx_ind,tauMF_0_ind,tauMF_ind);
695 //printf("Point to the right: (x,y)=(%Lf,%Lf) ;; (u,v)=(%Lf,%Lf)\nPoint above: (x,y)=(%Lf,%Lf) ;; (u,v)=(%Lf,%Lf)-\n",x_dir,y_dir,u_dir,v_dir,x_topo,y_topo,u_topo,v_topo);
696 }
697 else if(refineCheck==1)
698 {
699     if(fabs1(x_0)<0.0001)
700         x_est=0;
701     else
702         x_est=x_0;
703     if(fabs1(y_0)<0.0001)
704         y_est=0;
705     else
706         y_est=y_0;
707     inter_est=fopen("y*.txt","a");
708     fprintf(inter_est, "%Lf %Lf 0 0 0\n",x_est,y_est);
709     fclose(inter_est);
710     printf("\nStagnation point: (x,y)=(%Lf,%Lf)\n",x_est,y_est);
711     printf("--Interface analysis--\nThe number of interface points is: %d\nFor the Interface Representation run command:\npython3 graphv2.py &\n\n", inter_count);
712     printf("--Flow asymmetry parameter--\n");
713     printf("Relevant interface positions:\nLeft (x=%0.1Lf): y=%0.12Lf ;; Right (x=%0.1Lf): y=%0.12Lf\n",x_aux2,inter_aux2,x_aux,inter_aux);
714     q1=0.5*fabs1(inter_aux-y_est);
715     q2=1-q1;
716     dq=fabs1(q2-q1);
717     printf("q1 = %Lf ;; q2 = %Lf\n",q1,q2);
718     printf("|DQ| = %0.12Lf\n",dq);
719     printf("--Local Weissenberg Number--\n");
720     //1) Lambda
721     cp=fopen("../constant/constitutiveProperties", "r");

```

```

722 if (cp==NULL)
723 {
724     printf("Could not open constitutive properties, check file directory.\n\n");
725     return 0;
726 }
727 lambda_n=find_lambda_n(cp,word,lambda_n);
728 lambda_s=find_lambda_s(cp,word,lambda_s);
729 if((lambda_n!=-1) && (lambda_s!=-1))
730 {
731     printf("Lambda not found: There is no Weissenberg number in a fully newtonian crossing\n\n");
732     lambda_check=0;
733     numLambdas=0;
734 }
735 else if((lambda_n!=1) && (lambda_s!=-1))
736 {
737     printf("Just one of the fluids is viscoelastic\n");
738     printf("Lambda = %f\n",lambda_n);
739     numLambdas=1;
740 }
741 else if((lambda_n!=-1) && (lambda_s!=1))
742 {
743     printf("Both fluids are viscoelastic\n");
744     printf("Northern inlet Lambda = %f\nSouthern inlet Lambda = %f\n",lambda_n,lambda_s);
745     numLambdas=2;
746 }
747 else
748     printf("Something wrong with find_lambda\n");
749 // //-// wi_0 -> utilizes the 4 point closest to the stagnation point
750 lines=0;
751 rewind(fp);
752 for (ch=getc(fp); ch != EOF; ch=getc(fp))
753 {
754     if(ch=='\n')
755     {
756         comma=2;

```

```

757         lines++;
758         if(lines!=0)
759             fscanf(fp,"%Lf,%Lf",&u,&v);
760     }
761     else if(ch == ',')
762     {
763         comma++;
764         if((comma==alpha_ind) && (lines!=0))
765         {
766             fscanf(fp, "%Lf", &alpha1);
767         }
768         if((comma==x_ind) && (lines!=0))
769         {
770             fscanf(fp, "%Lf,%Lf", &x_wi, &y_wi);
771             if((fabsl(x_wi-x_est)<0.0001) && (y_wi>y_est)) //Cell center values above stagnation point
772             {
773                 if(y_wi<y_n4)
774                 {
775                     x_n4=x_wi;
776                     y_n4=y_wi;
777                     u_n4=u;
778                     v_n4=v;
779                 }
780             }
781             else if((fabsl(x_wi-x_est)<0.0001) && (y_wi<y_est)) //Cell center values bellow stagnation point
782             {
783                 if(y_wi>y_s4)
784                 {
785                     x_s4=x_wi;
786                     y_s4=y_wi;
787                     u_s4=u;
788                     v_s4=v;
789                 }
790             }
791             else if((fabsl(y_wi-y_est)<0.0001) && (x_wi>x_est)) //Cell center values to the right of the stagnation point

```

```

792         {
793             if(x_wi<x_e4)
794             {
795                 x_e4=x_wi;
796                 y_e4=y_wi;
797                 u_e4=u;
798                 v_e4=v;
799             }
800         }
801         else if((fabs(y_wi-y_est)<0.0001) && (x_wi<x_est)) //Cell center values to the left of the stagnation point
802         {
803             if(x_wi>x_o4)
804             {
805                 x_o4=x_wi;
806                 y_o4=y_wi;
807                 u_o4=u;
808                 v_o4=v;
809             }
810         }
811     }
812 }
813 }
814 dudx=(u_e4-u_o4)/(x_e4-x_o4);
815 dudy=(u_n4-u_s4)/(y_n4-y_s4);
816 dvdx=(v_e4-v_o4)/(x_e4-x_o4);
817
818 printf("Weissenberg calculations:\n");
819
820 if(numLambdas==1)
821     lambda=lambda_n;
822 if(numLambdas==2)
823 {
824     lambda=(lambda_n*(y_est-y_s4)+lambda_s*(y_n4-y_est))/(y_n4-y_s4);
825     printf("Lambda(3) = %f\n",lambda);
826 }

```

```

827
828 wi=lambda*sqrt(dudx*dudx+dvdx*dudy);
829 printf("wi0 = %Lf\n\n",wi);
830 printf("Point N from the stagnation point: (x,y)=(%Lf,%Lf) (u,v)=(%Lf,%Lf)\n",x_n4,y_n4,u_n4,v_n4);
831 printf("Point O from the stagnation point: (x,y)=(%Lf,%Lf) (u,v)=(%Lf,%Lf)\n",x_o4,y_o4,u_o4,v_o4);
832 printf("Point E from the stagnation point: (x,y)=(%Lf,%Lf) (u,v)=(%Lf,%Lf)\n",x_e4,y_e4,u_e4,v_e4);
833 printf("Point S from the stagnation point: (x,y)=(%Lf,%Lf) (u,v)=(%Lf,%Lf)\n\n",x_s4,y_s4,u_s4,v_s4);
834
835 fclose(fp);
836 }
837 //END OF MAIN
838 return 0;
839 }
840
841 float find_lambda_n(FILE *cp, const char *word, float lambda_n)
842 {
843     char str[BUFFER_SIZE];
844     char lambda_aux[5];
845     char *pos;
846     int indicador=0,i=0,j=0;
847
848     while ((fgets(str, BUFFER_SIZE, cp)) != NULL)
849     {
850         pos = strstr(str, word);
851
852         if (pos != NULL)
853         {
854             indicador++;
855         }
856         if((pos != NULL) && (indicador==1))
857         {
858             break;
859         }
860     }
861     // Not finding the word condition

```

```

862     if (indicador==0)
863         return -1;
864
865     //String cut
866     int len=strlen(str);
867     //Finding cutting position
868     for(i=0;i<(len-2);i++)
869     {
870         if(str[i]=='|')
871         {
872             indicador=i;
873             break;
874         }
875     }
876     i=0;
877     indicador++;
878     //Cut
879     for(i=0;i<(len-2);i++)
880     {
881         if(i>indicador)
882         {
883             lambda_aux[j]=str[i];
884             j++;
885         }
886     }
887     lambda_n=atof(lambda_aux);
888
889     return lambda_n;
890 }
891
892 float find_lambda_s(FILE *cp, const char *word, float lambda_s)
893 {
894     char str[BUFFER_SIZE];
895     char lambda_aux[5];
896     char *pos;|

```

```

897     int indicador=0,i=0,j=0;
898
899     while ((fgets(str, BUFFER_SIZE, cp)) != NULL)
900     {
901         pos = strstr(str, word);
902
903         if (pos != NULL)
904         {
905             indicador++;
906         }
907         if((pos != NULL) && (indicador==1))
908         {
909             break;
910         }
911     }
912     // Not finding word condition
913     if (indicador==0)
914         return -1;
915
916     //String cut
917     int len=strlen(str);
918     //Finding cut position
919     for(i=0;i<(len-2);i++)
920     {
921         if(str[i]==' ')
922         {
923             indicador=i;
924             break;
925         }
926     }
927     i=0;
928     indicador++;
929     //Cut
930     for(i=0;i<(len-2);i++)
931

```

```

932         if(i>indicador)
933         {
934             lambda_aux[j]=str[i];
935             j++;
936         }
937     }
938     lambda_s=atof(lambda_aux);
939
940     return lambda_s;

```


Appendix C

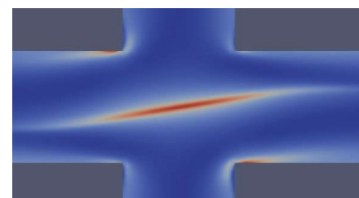
Powerpoint presentation

Numerical studies of purely elastic instabilities in
micro-geometries:
non-homogeneous concentration effects

Trabalho realizado por:
Francisco Martins



Orientado por:
Prof. Fernando Manuel Coutinho Tavares de Pinho
Prof. Alexandre Miguel Prior Afonso



Introdução

U. PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

RheoTool

- Solver utilizado é o **rheoInterFoam**
- Simula escoamentos de **fluidos viscoelásticos bifásicos**

Resposta elástica
Resposta viscosa

λ = Tempo de relaxação do fluido

Número de Débora

$$De = \lambda \frac{U}{D}$$

Número de Weissenberg local

$$Wi_0 = \lambda \dot{\epsilon}_0$$

Taxa de deformação no ponto de estagnação

$$\dot{\epsilon}_0 = \sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \frac{\partial u}{\partial y} \cdot \frac{\partial v}{\partial x}}$$

Introdução | Estudo de malha | UCM/Newtoniano | UCM/UCM | OdB/OdB | Conclusões | 1/18

Introdução

U. PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Escoamentos Newtonianos em cross-slot

[1] M. Davoodi, A. Domingues, and R. Poole. Control of a purely elastic symmetry-breaking flow instability in cross-slot geometries. *Journal of Fluid Mechanics*, 881:1123–1157, 10 2019.

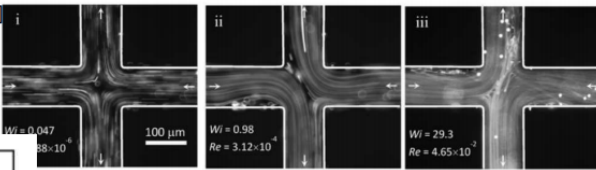
Newtonian

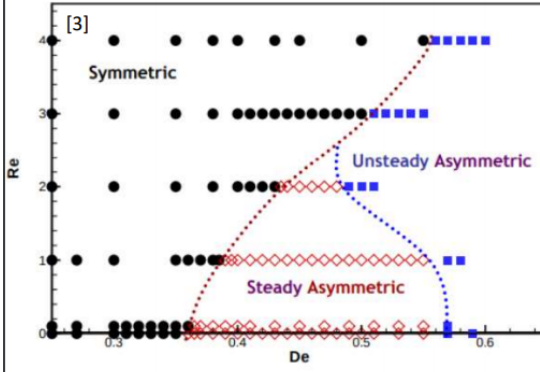
- $\xi = -1$, escoamento rotacional análogo a um sólido
- $\xi = 0$, escoamento de corte simples
- $\xi = 1$, escoamento puramente extensional

Introdução | Estudo de malha | UCM/Newtoniano | UCM/UCM | OdB/OdB | Conclusões | 2/18

Introdução

Escoamentos Viscoelásticos em cross-slot

[2] 

[3] 

[2] P. Sousa, F. Pinho, M. Oliveira, and M. Alves. Purely elastic flow instabilities in microscale cross-slot devices. *Soft matter*, 11, 09 2015.

[3] R. J. Poole, M. A. Alves, A. Afonso, F. T. Pinho, and P. J. Oliveira. Purely elastic flow instabilities in cross-slot flow. *The Society of Rheology 79th annual meeting*, 10 2007.

- Escoamento estacionário simétrico
- Escoamento estacionário assimétrico
- Escoamento instacionário

Wi_0

Introdução Estudo de malha UCM/Newtoniano UCM/UCM OdB/OdB Conclusões 3/18

Introdução

Concentrações não homogêneas

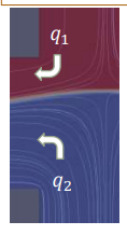
Número de Capilaridade

$$Ca = \frac{\mu v}{\sigma}$$

μ = viscosidade dinâmica
 v = velocidade característica
 σ = tensão superficial

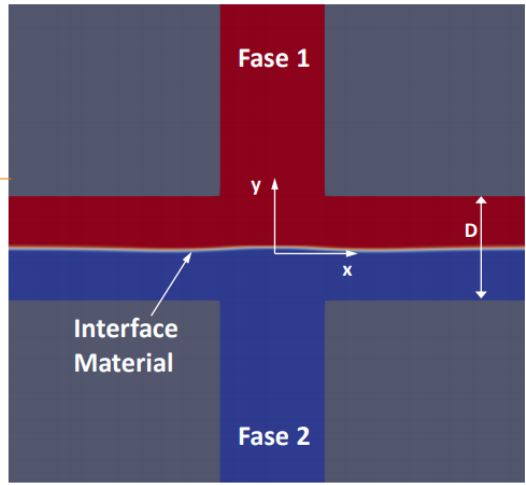
Parâmetro de assimetria do escoamento

$$DQ = \frac{q_2 - q_1}{q_1 + q_2}$$



Distância vertical normalizada do ponto de estagnação ao centro da geometria

$$y^* = \frac{y_{\text{Ponto de estagnação}}}{D}$$



Introdução Estudo de malha UCM/Newtoniano UCM/UCM OdB/OdB Conclusões 4/18

Estudo de Malha

U. PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Mesh	NP	NC in Block I	NC in Blocks II/III/IV/V	$\frac{\Delta t_{min}}{D} = \frac{\Delta y_{min}}{D}$
M1-NU	30368	2601	3060	0.02
M2-NU	120098	10404	12240	0.01
M3-NU	477650	41616	48960	0.005

NP = número de pontos ; **NC** = número de células

- Recriação de condições por Cruz et al. [4]
- Escoamento monofásico UCM
- Re=0.01 , $\sigma=0$, De=0.33
- 3 tipos de refinamento de malha não uniforme

Mesh	Wi0	Wi0 by Cruz et al.	DQ	t (s)	ϵ_1	ϵ_2
M1-NU	0.523	0.509	0.250	10^4	0.2%	20,8%
M2-NU	0.533	0.502	0.295	10^5	2.1%	6,6%
M3-NU	0.526	0.500	0.310	10^6	0.8%	1.8%
Extrapolated value	0.522	0.499	0.312	—	—	—

- t = ordem de tempo da simulação
- ϵ_1 =erro de Wi_0
- ϵ_2 =erro de |DQ|

[4] F. A. Cruz, R. J. Poole, A. M. Afonso, F. T. Pinho, P. J. Oliveira, and M. A. Alves. A new viscoelastic benchmark flow: Stationary bifurcation in a cross-slot. J. Non-Newtonian Fluid Mech 214, pages 57–68, 10 2014.

Introdução

Estudo de Malha

UCM/Newtoniano

UCM/UCM

OdB/OdB

Conclusões

5/18

UCM/Newtoniano

U. PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Re=0.01 / Re → 0

		De	Wi_0
Re=0.01	Ca=1	0.40	1.21
	Ca=∞	0.41	1.22
Re=0	Ca=1	0.44	1.32
	Ca=∞	0.45	1.32

$Re = Re_{UCM} = Re_N$

$Ca = Ca_{UCM} = Ca_N$

$Ca = \frac{\mu v}{\sigma}$

Re=1 ; Ca=∞

Deborah Number	Local Weissenberg Number
0.10	0.32
0.31	0.98
0.33	1.05
0.34	1.08
0.35	1.11
0.38	1.20
0.47	1.55
0.48	*
0.49	*
0.50	*
0.51	*
0.55	*
0.60	*
0.70	*

* = escoamento instacionário

Introdução

Estudo de malha

UCM/Newtoniano

UCM/UCM

OdB/OdB

Conclusões

6/18

Efeitos do número de capilaridade

• Para $Re_{UCM} = Re_N = 1$, avaliaram-se quatro valores de Débora próximos do crítico $De_1 = 0.47$

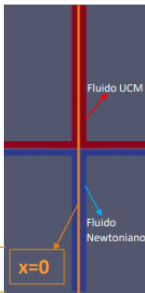
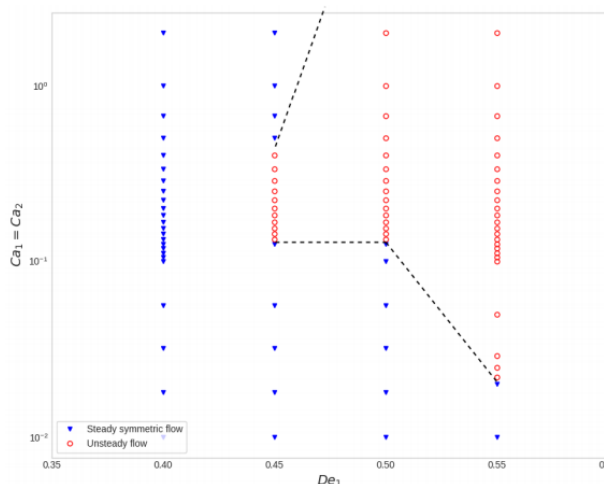
• Efeito estabilizante da tensão superficial

• 2 valores críticos para $De_1 = 0.45$:
 $Ca_1 = Ca_2 = 0.125$ e
 $Ca_1 = Ca_2 = 0.500$

↓

• Dualidade de efeitos perto de condições críticas

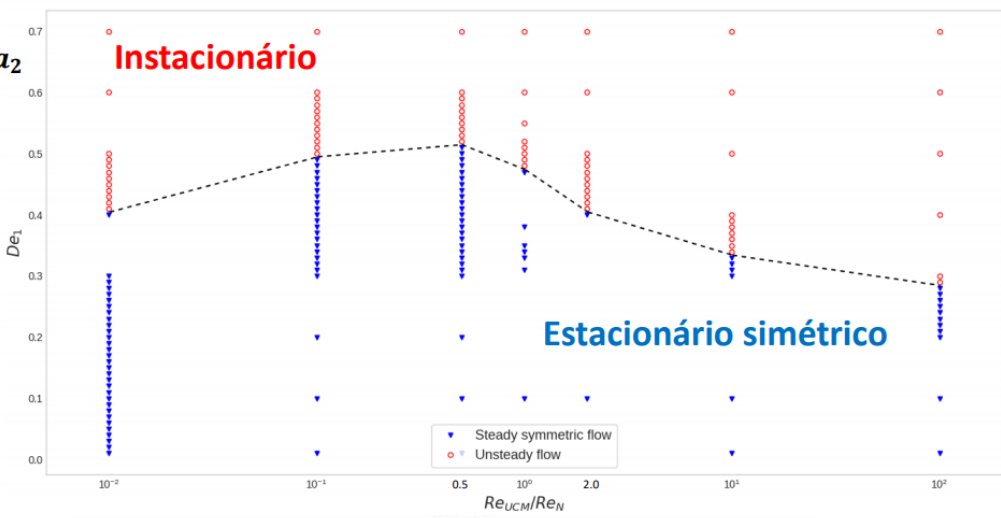
$De_1 = 0.45$	
$(Ca_1 = Ca_2) \geq 0.500$	$(Ca_1 = Ca_2) \leq 0.125$
$\Sigma\tau_{yy} = -42$ [Pa]	$\Sigma\tau_{yy} = -43$ [Pa]

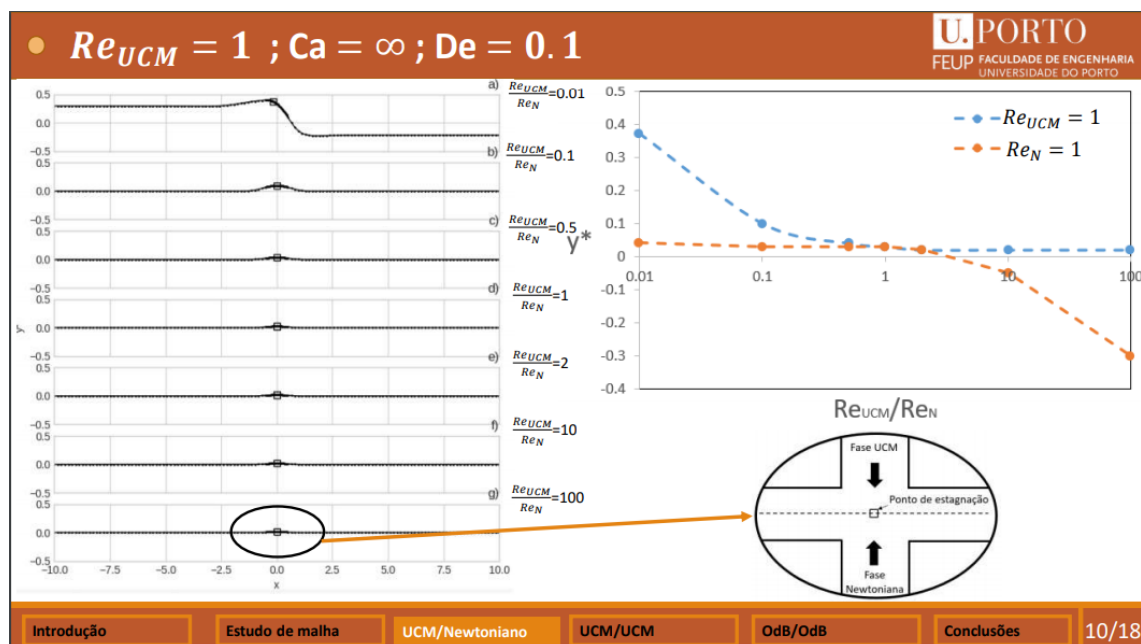
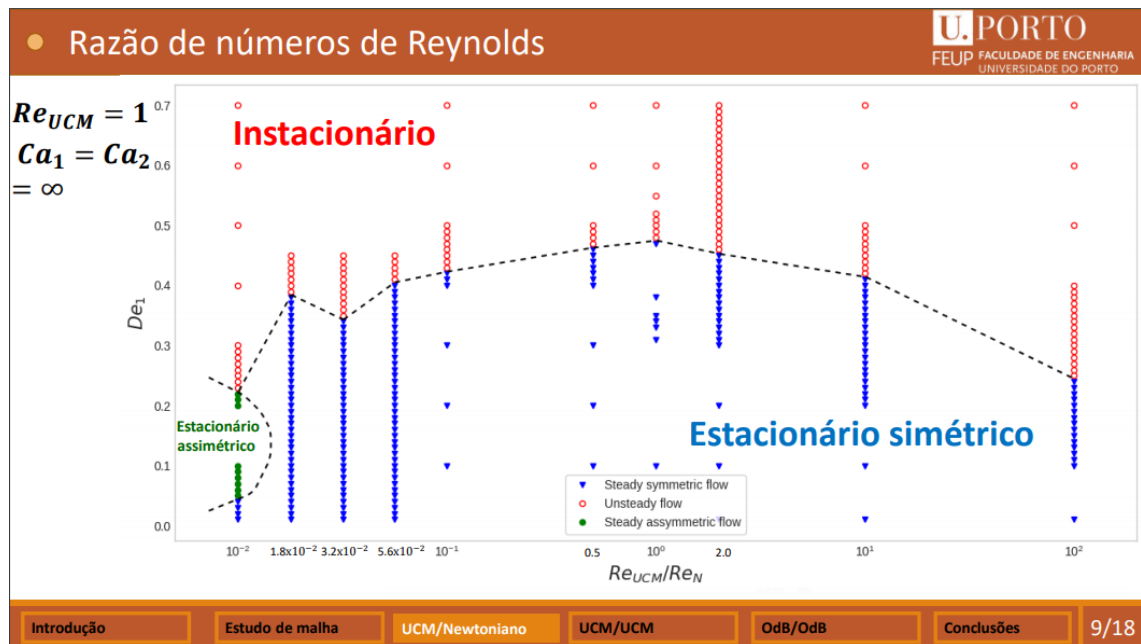
Introdução Estudo de malha UCM/Newtoniano UCM/UCM OdB/OdB Conclusões 7/18

Razão de números de Reynolds

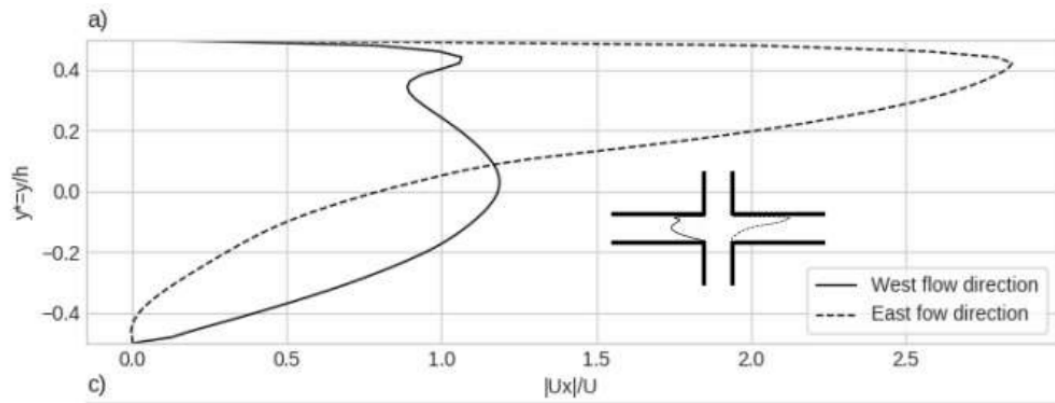
$Re_N = 1$
 $Ca_1 = Ca_2$
 ∞



Introdução Estudo de malha UCM/Newtoniano UCM/UCM OdB/OdB Conclusões 8/18



Identificação de escoamentos assimétricos



Introdução

Estudo de malha

UCM/Newtoniano

UCM/UCM

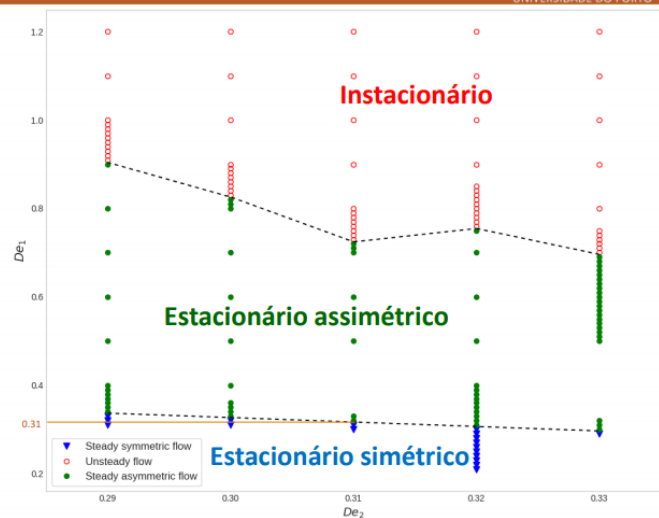
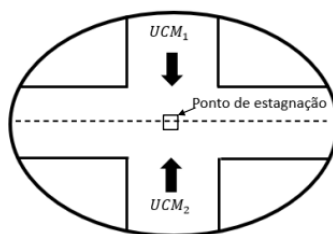
OdB/OdB

Conclusões

11/18

UCM/UCM

- De_1 e De_2 representam o número de Débora da fase superior e inferior, respetivamente
- $Re_1 = Re_2 = 0.01$ e $Ca_1 = Ca_2 = \infty$



Introdução

Estudo de malha

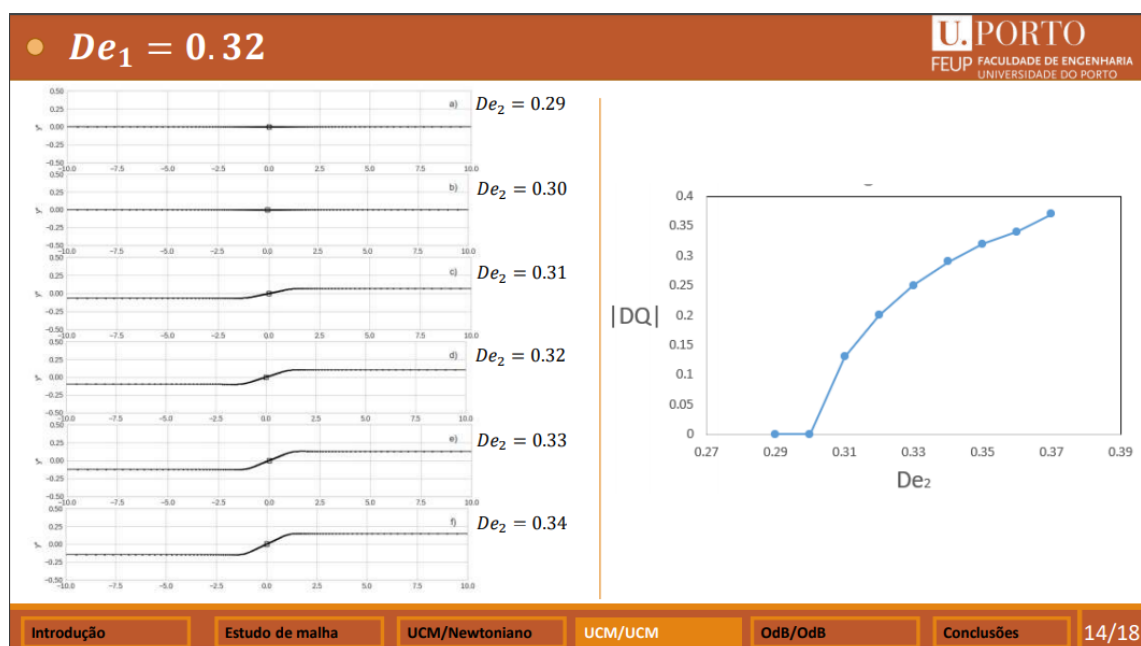
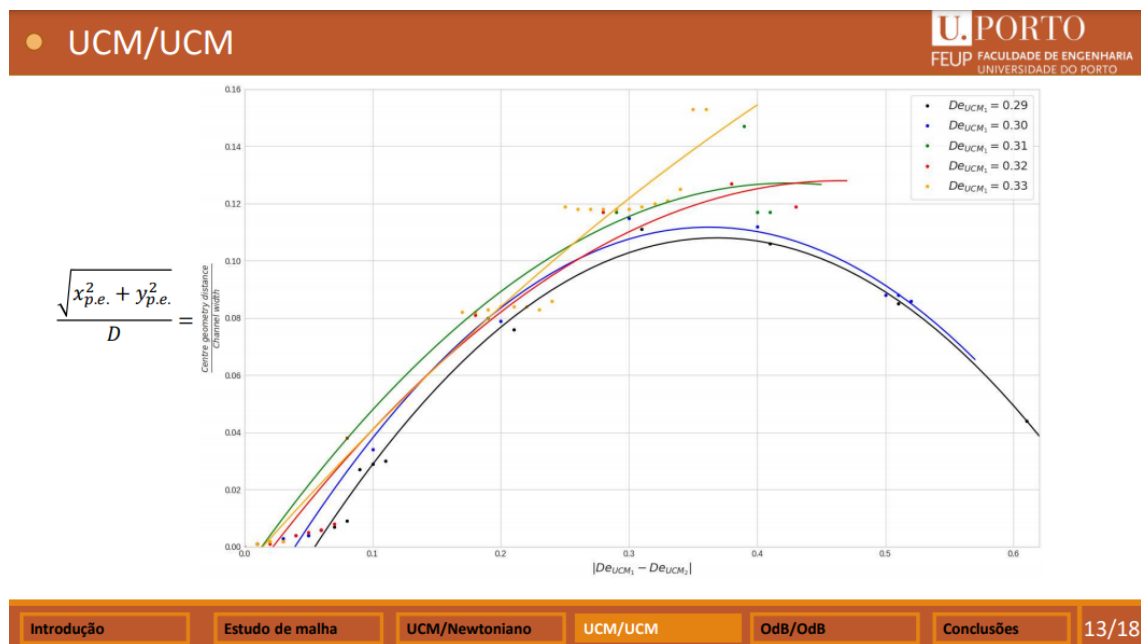
UCM/Newtoniano

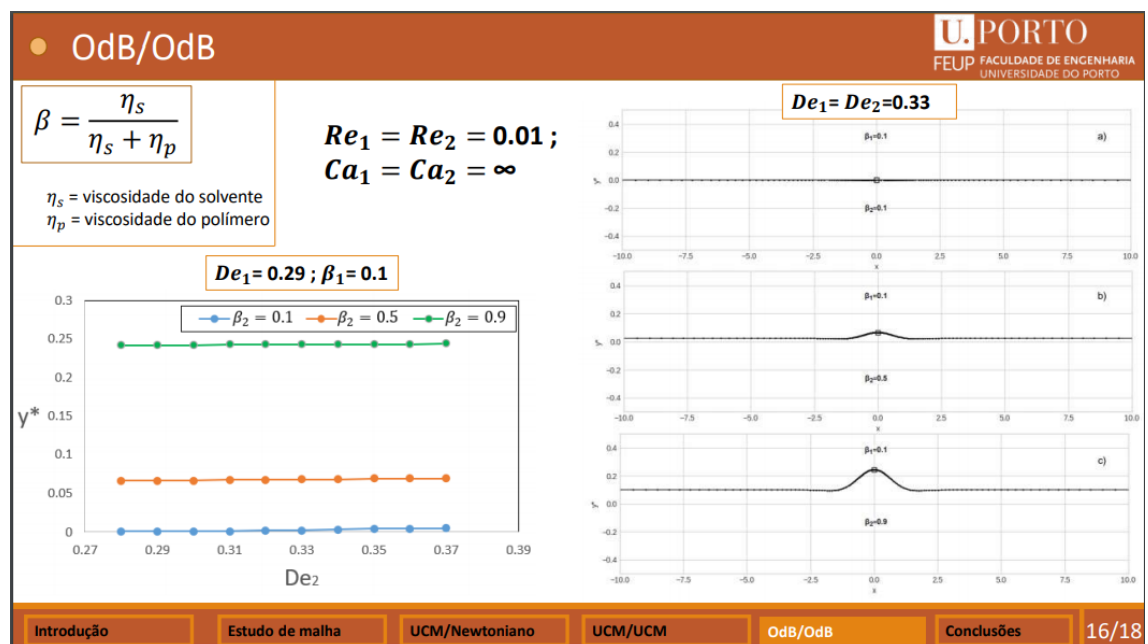
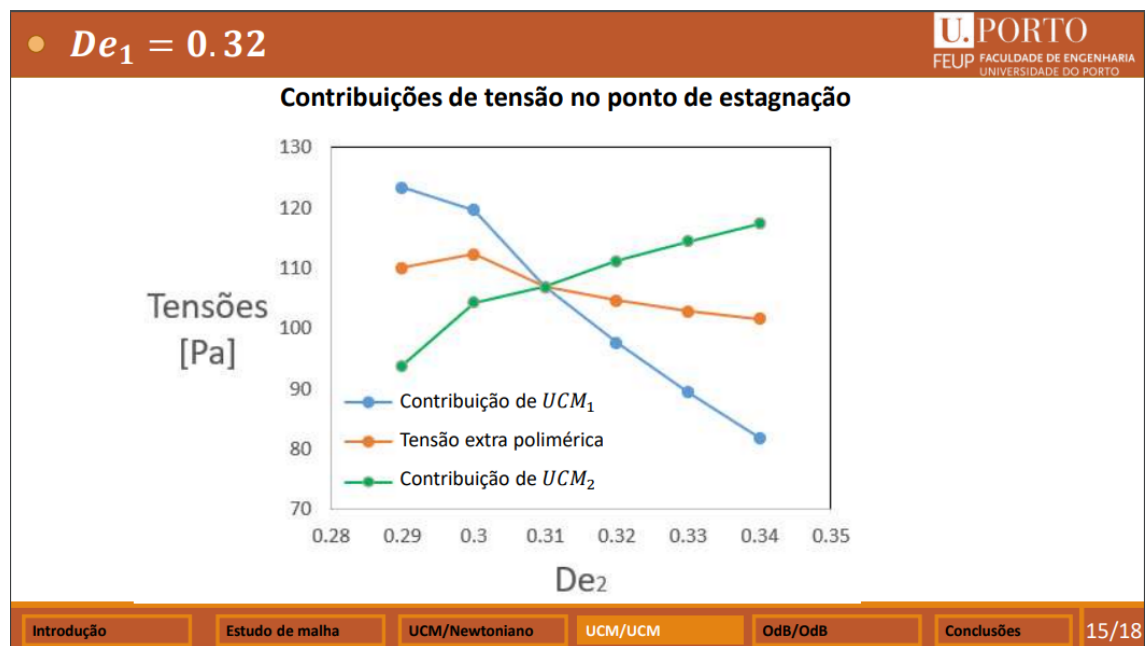
UCM/UCM

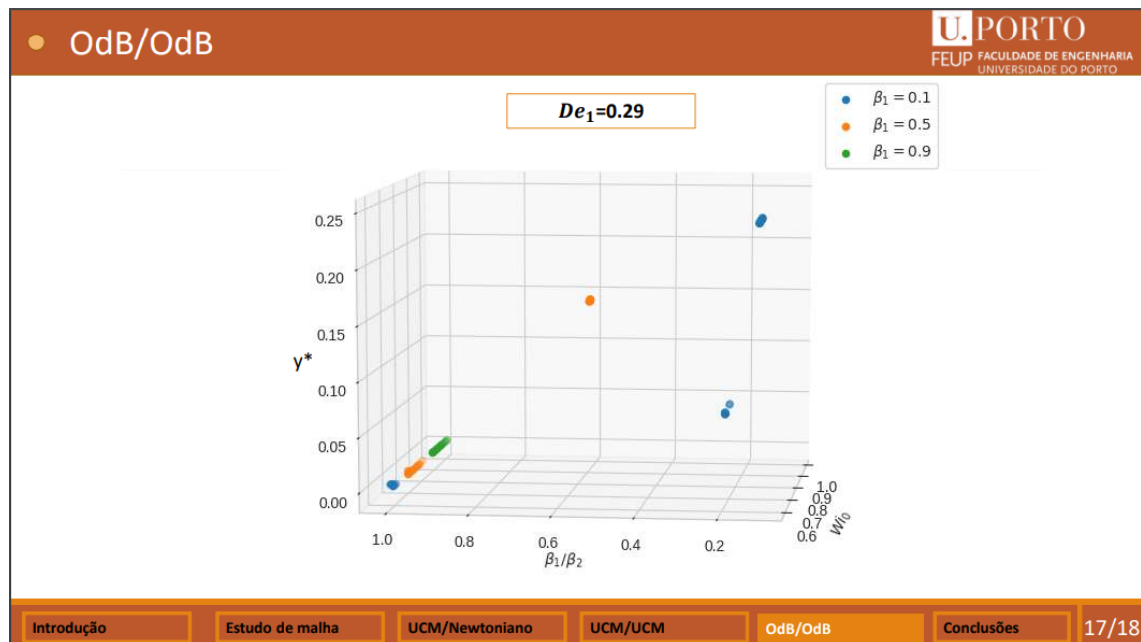
OdB/OdB

Conclusões

12/18







Conclusões e trabalhos futuros

U. PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

UCM/Newtoniano

- Apenas dois regimes de escoamento foram encontrados para a maioria dos casos
- Estudo do número de capilaridade revela potencial efeito estabilizante perto de condições críticas
- Razão dos números de Reynolds entre fases demonstra que para as razões unitárias, o escoamento é estacionário até números de Débora mais elevados
- Números de Weissenberg locais acompanham os números de Débora do escoamento
- Ponto de estagnação encontra-se verticalmente desfasado do centro da geometria quando $\frac{Re_{UCM}}{Re_N} = 1$

UCM/UCM

- Três tipos de regime de escoamento encontrados
- Correspondência com valores monofásicos verificada
- Números de Weissenberg locais aparentam acompanhar o desfasamento do ponto de estagnação
- Assimetria do escoamento aumenta com o aumento do número de Débora de uma das fases

OdB/OdB

- Efeito estabilizador do solvente adicionado verificado
- Influência do número de Déborah muito baixa
- Ponto de estagnação e número de Weissenberg local reagem principalmente ao β do escoamento

Introdução Estudo de malha UCM/Newtoniano UCM/UCM OdB/OdB Conclusões 18/18

References

- [1] M.A. Alves, P.J. Oliveira, and F.T. Pinho. Numerical methods for viscoelastic fluid flows. *Annual Review of Fluid Mechanics*, 53(1):509–541, 2021.
- [2] R. B. Bird, R. C. Armstrong, and O. Hassager. Dynamics of polymeric liquids, volume 1: Fluid mechanics, 2nd edition. 1987.
- [3] A. H. D. Cheng and D. T. Cheng. Heritage and early history of the boundary element method. *Engineering Analysis with Boundary Elements*, 29(3):268–302, 2005.
- [4] T. Chinyoka, Y.Y. Renardy, M. Renardy, and D.B. Khismatullin. Two-dimensional study of drop deformation under simple shear for oldroyd-b liquids. *Journal of Non-Newtonian Fluid Mechanics*, 130(1):45–56, 2005.
- [5] R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, 11(2):215–234, 1967.
- [6] F. A. Cruz, R. J. Poole, A. M. Afonso, F. T. Pinho, P. J. Oliveira, and M. A. Alves. A new viscoelastic benchmark flow: Stationary bifurcation in a cross-slot. *J. Non-Newtonian Fluid Mech* 214, pages 57–68, 2014.
- [7] M. Davoodi, A. Domingues, and R. Poole. Control of a purely elastic symmetry-breaking flow instability in cross-slot geometries. *Journal of Fluid Mechanics*, 881:1123–1157, 2019.
- [8] R.A. Figueiredo, C.M. Oishi, A.M. Afonso, I.V.M. Tasso, and J.A. Cuminato. A two-phase solver for complex fluids: Studies of the weissenberg effect. *International Journal of Multi-phase Flow*, 84:98–115, 2016.
- [9] J. L. Fávero. Simulação de escoamentos viscoelásticos: desenvolvimento de uma metodologia de análise utilizando o software openfoam e equações constitutivas diferenciais. 2009.
- [10] C. J. Greenshields. Openfoam user guide, volume 4. 2016.
- [11] H. Jeffreys. The earth; its origin, history and physical constitution. *Quarterly Journal of the Royal Meteorological Society*, 51(213):64–65, 1925.
- [12] J. S. Lee, R. Dylla-Spears, N. P. Teclemariam, and S. J. Muller. Microfluidic four-roll mill for all flow types. *Applied Physics Letters*, 90(7):074103, 2007.
- [13] J. C. Maxwell. On the dynamical theory of gases. pages 197–261.
- [14] J. G. Oldroyd and A. H. Wilson. On the formulation of rheological equations of state. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 200(1063):523–541, 1950.

- [15] M.S.N. Oliveira, F.T. Pinho, R.J. Poole, P.J. Oliveira, and M.A. Alves. Purely elastic flow asymmetries in flow-focusing devices. *Journal of Non-Newtonian Fluid Mechanics*, 160(1):31–39, 2009. Complex flows of complex fluids.
- [16] F. Pimenta and M. A. Alves. Stabilization of an open-source finite-volume solver for viscoelastic fluid flows. *J. Non-Newtonian Fluid Mech* 239, pages 85–104, 2017.
- [17] F. Pimenta and M. A. Alves. Rheotool user guide. 2020.
- [18] R. J. Poole, M. A. Alves, A. Afonso, F. T. Pinho, and P. J. Oliveira. Purely elastic flow instabilities in cross-slot flow. *The Society of Rheology 79th annual meeting*, 2007.
- [19] R. J. Poole, M. A. Alves, and P. J. Oliveira. Purely elastic flow asymmetries. *Phys. Rev. Lett.*, 99:164503, 2007.
- [20] R. J. Poole, G. N. Rocha, and P. J. Oliveira. A symmetry-breaking inertial bifurcation in a cross-slot flow. *Computers & Fluids*, 93:91–99, 2014.
- [21] O. Shonibare. Numerical simulation of viscoelastic multiphase flows using an improved two-phase flow solver. *Open Access Dissertation, Michigan Technological University*, 2017.
- [22] A. Sidi. Generalizations of richardson extrapolation with applications to numerical integration. *International Series of Numerical Mathematics*, 1988.
- [23] H. J. Simon, M. S. N. Oliveira, M. A. Alves, and G. H. McKinley. Optimized cross-slot flow geometry for microfluidic extensional rheometry. *Phys. Rev. Lett.*, 109:128301, 2012.
- [24] M. Sommerfeld. Numerical methods for dispersed multiphase flows. *Advances in Mathematical Fluid Mechanics*, pages 327–396, 2017.
- [25] J. Soulages, M. Oliveira, P. Sousa, M. Alves, and G. Mckinley. Investigating the stability of viscoelastic stagnation flows in t-shaped microchannels. *Journal of Non-Newtonian Fluid Mechanics*, 163:9–24, 2009.
- [26] P. Sousa, F. Pinho, M. Oliveira, and M. Alves. Purely elastic flow instabilities in microscale cross-slot devices. *Soft matter*, 11, 2015.