

A High-Order (EXTended) Discontinuous Galerkin Solver for the Simulation of Viscoelastic Droplet

Kikker, Anne
(2020)

DOI (TUprints): <https://doi.org/10.25534/tuprints-00012308>

Lizenz:



CC-BY-SA 4.0 International - Creative Commons, Attribution Share-alike

Publikationstyp: Ph.D. Thesis

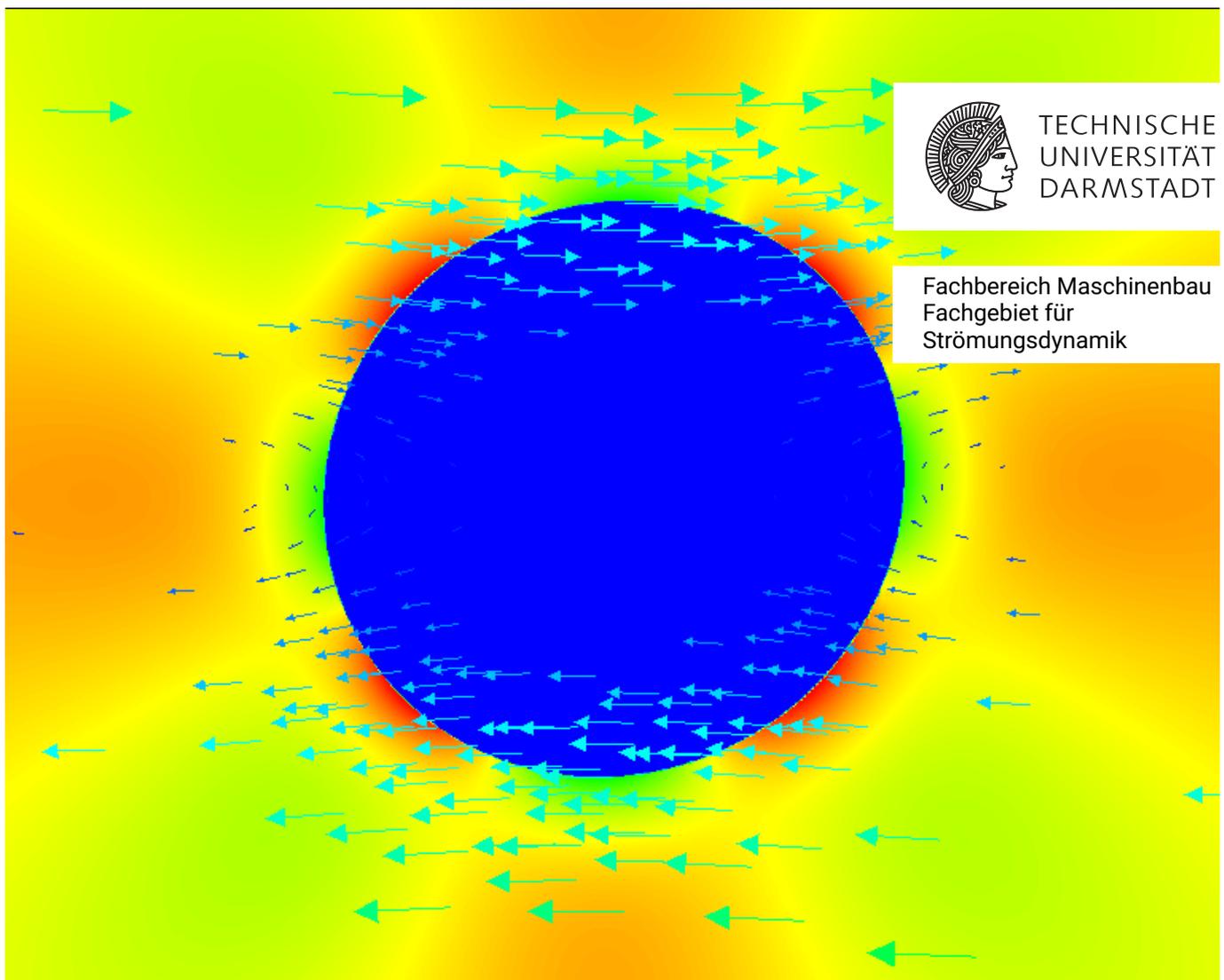
Fachbereich: 16 Department of Mechanical Engineering

Quelle des Originals: <https://tuprints.ulb.tu-darmstadt.de/12308>

A High-Order (EXtended) Discontinuous Galerkin Solver for the Simulation of Viscoelastic Droplet

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
Genehmigte Dissertation von Anne Kikker aus Trier
Tag der Einreichung: 12. April 2020, Tag der Prüfung: 10. Juni 2020

1. Gutachten: Prof. Dr.-Ing. M. Oberlack
2. Gutachten: Prof. Dr. rer. nat. D. Bothe
Darmstadt



A High-Order (EXtended) Discontinuous Galerkin Solver for the Simulation of Viscoelastic Droplet

Genehmigte Dissertation von Anne Kikker

1. Gutachten: Prof. Dr.-Ing. M. Oberlack
2. Gutachten: Prof. Dr. rer. nat. D. Bothe

Tag der Einreichung: 12. April 2020

Tag der Prüfung: 10. Juni 2020

Darmstadt

Bitte zitieren Sie dieses Dokument als:

URN: urn:nbn:de:tuda-tuprints-123084

URL: <http://tuprints.ulb.tu-darmstadt.de/12308>

Dieses Dokument wird bereitgestellt von tuprints,

E-Publishing-Service der TU Darmstadt

<http://tuprints.ulb.tu-darmstadt.de>

tuprints@ulb.tu-darmstadt.de

Die Veröffentlichung steht unter folgender Creative Commons Lizenz:

CC BY-SA 4.0 (Namensnennung – Weitergabe unter gleichen Bedingungen 4.0 International)

<https://creativecommons.org/licenses/by-sa/4.0/>

Difficult problems are seldom endowed with miracle solutions.

M.J. Crochet, A.R. Davies, K. Walters (1984): Numerical Simulation of Non-Newtonian Flow.

Erklärungen laut Promotionsordnung

§8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 12. April 2020

A.Kikker

Abstract

The aim of this work is to provide a solver for viscoelastic multi-phase flows within the Bounded Support Spectral Solver (BoSSS) currently under development at the Chair of Fluid Dynamics at the Technical University of Darmstadt. The discretisation in BoSSS consists of a high-order Discontinuous Galerkin (DG) method for single-phase flow and a high-order eXtended Discontinuous Galerkin (XDG) method for the multi-phase purpose. The solver shall be used to investigate numerically the behaviour of viscoelastic droplets. The macroscopic Oldroyd B model which is used in a wide range of applications is chosen as the constitutive model. A detailed derivation of the system of equations including the modeling principles for the Oldroyd B model is presented.

A DG discretisation of the system of equations including the Local Discontinuous Galerkin (LDG) method is presented after introducing the field of the DG method. The derivation of appropriate flux functions for the constitutive equations and the extra stress tensor are one of the key derivations of this scientific work.

Difficulties arising in the numerical solution of viscoelastic flow problems for higher Weissenberg numbers for different discretisation methods are due to the convection dominated, mixed hyperbolic-elliptic-parabolic nature of the system of equations. Several strategies are presented which overcome these problems and are known from the literature. A key achievement of this scientific work is the application of the LDG method, originally developed for a hyperbolic system of equations for a Newtonian fluid, on the viscoelastic system of equations which renders methods for preserving ellipticity unnecessary. Furthermore, various strategies to enhance and to support convergence of the solution of the DG discretised system are presented. These are the Newton method with different approaches determining the Jacobian of the system, a homotopy continuation method based on the Weissenberg number for a better initial guess for the Newton method, and a troubled cell indicator combined with an artificial diffusion approach or an adaptive mesh refinement strategy, respectively.

For the completeness of this work the XDG method is presented using a sharp interface approach with a signed distance level-set function as it is implemented in BoSSS. The single-phase solver is combined with these methods and appropriate flux functions for the interface are implemented to enable multi-phase applications for viscoelastic fluid.

Several numerical experiments are conducted to verify and to validate the viscoelastic single-phase solver and to show the capability of the viscoelastic multi-phase solver to simulate viscoelastic droplets. Advantages and disadvantages of the implementation and an outlook for future research can be found in the conclusion.

Zusammenfassung

Als Ziel der vorliegenden Arbeit soll ein Löser für viskoelastische Mehrphasenströmungen zur Verfügung gestellt werden. Dieser Löser ist eingebettet in die Software Bounded Support Spectral Solver (BoSSS), die aktuell am Fachgebiet für Strömungsdynamik an der Technischen Universität Darmstadt entwickelt wird. Die in BoSSS verwendete Diskretisierung ist eine diskontinuierliche Galerkin (DG) Methode hoher Ordnung für einphasige Strömungen und eine erweiterte (eXtended) DG (XDG) Methode hoher Ordnung für den mehrphasigen Anwendungsfall. Der Löser soll dazu genutzt werden, das Verhalten viskoelastischer Tropfen numerisch zu untersuchen. Als Konstitutivgleichung wird das makroskopische Oldroyd B Modell, das in vielen Bereichen Anwendung findet, ausgewählt. Die Herleitung des zugrundeliegenden Gleichungssystems inklusive der Prinzipien zur Modellierung des Oldroyd B Modells werden gezeigt.

Die DG Diskretisierung des Gleichungssystems und die lokale DG (LDG) Methode werden inklusive einer Einführung in die DG Methodik präsentiert. Die Ableitung geeigneter Flussformulierungen für die Konstitutivgleichungen ist dabei eines der zentralen Ergebnisse der vorliegenden wissenschaftlichen Arbeit.

Aufgrund des konvektionsdominierten, gemischt hyperbolisch-elliptisch-parabolischen Gleichungssystems können Probleme bezüglich der numerischen Lösung des viskoelastischen Gleichungssystems für höhere Weissenbergzahlen für verschiedene Diskretisierungsmethoden entstehen. Einige Ansätze zur Lösung dieser Probleme, die in der Literatur beschrieben werden, werden präsentiert. Ein weiteres zentrales Ergebnis dieser wissenschaftlichen Arbeit ist der Einsatz der LDG Methode, die ursprünglich für hyperbolische Gleichungssysteme für ein newtonsches Fluid entwickelt wurde und deren Anwendung Methoden zur Erhaltung elliptischer Eigenschaften im Gleichungssystem unnötig macht. Des Weiteren werden einige Strategien vorgestellt, deren Ziel die Verbesserung der Konvergenz der Lösung für das diskretisierte Gleichungssystem ist. Diese sind das Newtonverfahren mit unterschiedlichen Ansätzen zur Ermittlung der Jacobi-Matrix des Systems, eine Homotopie-Fortsetzungsmethode basierend auf der Weissenbergzahl für eine bessere Initiallösung im Newtonverfahren, und ein Indikator für gestörte Zellen kombiniert mit einer künstlichen Diffusion oder einer adaptiven Gitterverfeinerung.

Zur Vollständigkeit der Arbeit wird die XDG Methode, die in BoSSS implementiert ist, erläutert. Hierbei wird die Phasengrenze mittels einer vorzeichenbasierten Abstandsfunktion, der Level-Set-Funktion, dargestellt. Der Einphasenlöser wird mit diesen Methoden kombiniert und geeignete Flussfunktionen für die Phasengrenze werden implementiert, um eine Anwendung für viskoelastische Mehrphasenströmungen zu ermöglichen.

Einige numerische Ergebnisse des einphasigen Löses zur Verifikation und Validierung der Implementierung und des mehrphasigen Löses zur Machbarkeit zur Simulation von viskoelastischen Tropfen werden gezeigt. Vor- und Nachteile der Implementierung sowie ein Ausblick

für zukünftige Forschung können in der Schlussfolgerung nachgelesen werden.

Acknowledgements

First of all, I want to thank my supervisor Prof. Oberlack for all the fruitful discussions and important advises concerning the scientific work and also Prof. Bothe being my co-supervisor as part of the Graduate School of Computational Engineering having always helpful suggestions on the assemblies.

I also want to thank Florian for sharing all his expert knowledge as research group leader of the BoSSS working group and his persistent efforts getting my solver to run properly.

Of course my thanks go to all my colleagues at the Chair of Fluid Dynamics for all the support both in the scientific field and private discussions. Many became friends in the recent years and I acknowledge our good and supportive team spirit.

A special thanks goes to my good friend and office colleague Martin for enduring all the ups and downs with me, feeding me with chocolate when necessary, and for all his advise concerning the multi-phase solver. He really is an expert on this field.

I also thank my parents and sisters supporting and advising me in the recent years, always having some encouraging words for me.

And finally I want to thank my family who supported me so much in the last couple of month. I thank my daughter Enya who was so sympathetic being many hours in kindergarden, and my husband who took so much care for her beside his job. Especially in the last weeks we grew together in self-quarantine because of the Corona-virus spreading the world.

Reconciling a child and a PhD-thesis - many women on this world managed that before. But reconciling a child, a PhD-thesis and the Corona-crisis was the biggest challenge of my life so far.

Contents

List of symbols	xxi
List of Abbreviations	xxvii
1. Introduction	1
1.1. Physical Properties of Viscoelastic Material	2
1.2. Physical Properties of Droplets	3
1.3. Motivation and Objectives of this Work	6
2. Governing Equations	11
2.1. Continuum Mechanical Definitions	11
2.2. Conservation of Mass	13
2.3. Conservation of Momentum	13
2.4. Rheological Modelling	14
2.4.1. Inviscid and Newtonian Fluids	14
2.4.2. Principles in Rheological Modelling	15
2.4.3. Normal Stress Differences and Fading Memory	17
2.4.4. Viscoelastic Models	17
2.5. System of Equations for Viscoelastic Single-Phase Flow	20
2.5.1. Splitting into Polymeric and Solvent Part	20
2.5.2. Non-dimensionalisation	21
2.6. The Viscoelastic Multi-Phase Setting	23
3. The High Weissenberg Number Problem	27
3.1. Mathematical Aspects of Viscoelastic Models	28
3.2. Overview over Discretization Methods	29
3.3. Stabilization Methods for the Coupled System	32
3.3.1. Explicitly Elliptic Momentum Equation Method	33
3.3.2. Elastic Viscous Stress Splitting Type Discretization	34
3.3.3. Log-Conformation Formulation	35
3.4. Stabilization Methods for the Constitutive Equations	36
3.4.1. Streamline Upwinding (Petrov-Galerkin) Methods	36
3.4.2. Discontinuous Galerkin Formulation with Upwinding	37
4. The Discontinuous Galerkin Discretization for Viscoelastic Single-Phase Flow	39
4.1. State of Knowledge	39
4.2. Definitions for the Discontinuous Galerkin Space	41
4.3. Introduction to the Discontinuous Galerkin Discretization	42

4.4.	The Local Discontinuous Galerkin Method	45
4.5.	Time Discretization	46
4.6.	Discretization of the Viscoelastic Governing Equations	47
4.6.1.	Time Discretization using a Backward Difference Scheme	49
4.6.2.	Spatial Discretization using Local Discontinuous Galerkin	49
5.	The EXTENDED Discontinuous Galerkin Method for Viscoelastic Droplets	53
5.1.	State of Knowledge	53
5.2.	Definitions for the Extended Discontinuous Galerkin Space	53
5.3.	Interface Representation and Evolution	55
5.3.1.	Elliptic Re-initialisation	56
5.3.2.	The Extensional Problem	57
5.4.	Time Integration with a Moving Interface	58
5.5.	Spatial Discretization in Cut-Cells	59
5.5.1.	Cell Agglomeration	60
5.5.2.	Implicit High-Order Quadrature for Curved Surfaces	61
5.5.3.	Adaptive Mesh Refinement at the Interface	63
5.6.	Numerical Fluxes for the Two-Phase Setting	65
6.	Solution Strategies for the Algebraic Equation System	69
6.1.	Solver Structure	69
6.2.	Linearization of the System with the Newton method	70
6.2.1.	Implementation of the Newton Method	72
6.2.2.	Choice of the Approximation of the Jacobian	74
6.2.3.	Incremental Increase of the Weissenberg Number	75
6.3.	Convergence Supporting Strategies	76
6.3.1.	Troubled Cell Indicator	76
6.3.2.	Artificial Diffusion	77
6.3.3.	Adaptive Mesh Refinement for Troubled Cells	77
7.	Numerical Results	81
7.1.	Verification of the Local Discontinuous Galerkin Method	81
7.2.	Validation by the Confined Cylinder Benchmark Problem	82
7.2.1.	Convergence Study	85
7.2.2.	Results for Steady Flow Simulation	86
7.2.3.	Results for Unsteady Flow Simulation	90
7.2.4.	Results for Different Reynolds Numbers	90
7.3.	Droplet in Shear Flow	96
8.	Conclusion	101
	Bibliography	103
A.	BoSSSpad-Worksheets	113
A.1.	Validation Local Discontinuous Galerkin Implementation	113
A.2.	Confined Cylinder Benchmark	118
A.3.	Droplet in Shear Flow	123

List of Figures

1.1.	Dependency of the shear stress on the shear rate. The function displays the viscosity of the fluid for Newtonian, shear-thinning and shear-thickening (dilatant) behaviour.	3
1.2.	Different effects in viscoelastic fluids due to normal stress differences and a fading memory.	4
1.3.	Strain retardation and stress relaxation. On the left side a constant stress is applied to the viscoelastic fluid in the lower graph. The strain response is shown in the upper graph displaying a retardant behaviour slowly saturating to a final shear rate. At time t_0 a constant shear rate is applied to the fluid causing a relaxing stress response until the final stress is reached.	5
1.4.	Schematic representation of a molecule within a liquid drop in a gaseous ambient medium and one molecule at the surface of the drop. The arrows indicate the cohesion energy with the surrounding molecules. It can be seen that this energy is roughly halved at the surface leading to a tension at the surface.	6
1.5.	Schematic representation of the structure of the BoSSS solver from the BoSSS handbook (Kummer et al., 2020a).	9
2.1.	Pipkin diagram showing the relation between Weissenberg and Deborah number (Phan-Thien and Mai-Duy, 2017). The regions beyond the dashed lines show linear behaviour: For low Deborah numbers this is viscometric flow, for high Deborah numbers this is rubber-like elasticity. For low Weissenberg numbers the viscoelastic behaviour is linear for all Deborah numbers resulting in linear elasticity if $De \rightarrow \infty$. If both numbers are zero, the flow is Newtonian.	23
4.1.	A-stability regions of the Backward Differentiation Formula (BDF) $_z$ schemes for $1 \leq z \leq 4$. The inner regions of the circles are unstable, the outer regions are stable. It can be seen that the schemes up to the BDF2 are always stable for all $\Re(\lambda\Delta t) < 0$. For the BDF4 the opening angle of the $A(\theta)$ -stability region is marked.	48
5.1.	Sketch of a space-time cut-cell $K_{j,s}^*$ for a constant interface velocity u^* (Kummer et al., 2018).	60
5.2.	Cell Agglomeration in a two-dimensional equidistant Cartesian grid. The blue curve is the interface with $\varphi = 0$ cutting the cells in which it is located. If a cut-cell is smaller than a threshold a (red cells, left) it is agglomerated to its largest neighbour (green cells, right).	62

5.3.	Definition of Saye (2015) for the height function $\tilde{h}(\tilde{x})$ describing the zero level-set, here for the case that $\frac{\partial \varphi}{\partial x_i} < 0$ means it is beneath the graph of the height function, \tilde{x} abstractly represents a $d - 1$ -dimensional space.	64
5.4.	Recursive strategy for the search of the quadrature points for a Gaussian quadrature for a single two-dimensional cell. The red arrows follow a search direction until a root of the level-set is reached and the quadrature nodes are distributed along their vector depending on the order of quadrature.	64
5.5.	Curvature refined adaptive mesh refinement along an interface for a droplet. The amount of refinement levels is 2.	65
6.1.	Flowchart of the solver within a steady or the first time-step with different levels of iterative loops. The item 'solve' contains the linearisation scheme and the direct solver for the linearised system. In case of multi-phase flow it also includes the movement of the level-set. ϵ is the maximum artificial viscosity used and ϵ_0 the user defined starting value for ϵ . In unsteady simulations the Weissenberg number is fixed after the first time-step with Wi_{aim}	71
6.2.	Artificial diffusion using the troubled cell indicator presented for the refinement strategy. Since in this representation only qualitative values of the quantities are of interest, scale bars are omitted for simplification. Red colored cells mark high values, blue colored cells mark low values. in case of the troubled cell indicator the upper and lower bound s_0 are also used in the scaling of the coloring such that deep red and deep blue cells respectively, show cells with a troubled cell indicator out of the bounds s_0	78
6.3.	Adaptive mesh refinement using the troubled cell indicator presented for the refinement strategy. Since in this representation only qualitative values of the quantities are of interest, scale bars are omitted for simplification. Red colored cells mark high values, blue colored cells mark low values.	80
7.1.	h^k -convergence for the L_2 -error against the analytical solution in Eq. (7.1) up to a polynomial order of $k = 5$ for the velocity \mathbf{u} and the stresses $\boldsymbol{\tau}$ and of $k = 4$ for the pressure p . The solid lines mark the expected order of convergence of $k + 1$ for \mathbf{u} and p and k for $\boldsymbol{\tau}$	83
7.2.	Computational domain with Boundary Condition (BC) for velocities.	85
7.3.	Mesh mesh_1 with location of number of nodes as referred to in table 7.3.	85
7.4.	Convergence study for $Wi = 0$ (left) and $Wi = 0.2$ (right) in the L_2 -Norm compared to the Degrees of Freedom (DOF) of the finest mesh. The solid lines show the expected convergence rates with $k + 1$ for velocity and pressure and k for the stresses.	87
7.5.	Dimensionless drag force of the confined cylinder for different Weissenberg numbers for steady calculations (+) compared with unsteady calculations from exemplary literature (Kim et al., 2004; Keith et al., 2017; Claus and Phillips, 2013). For $Wi \leq 0.6$ we have good agreement. Afterwards, unsteady effects cause errors in the unstable steady state solution as expected.	88
7.6.	Elevate plots of the stress and velocity profiles over the domain $-5 \leq x \leq 5$ for $Wi = 0.3$. The acceleration of the fluid and the high stress peaks in the narrow at the cusp of the cylinder as well as the stress peak in the wake of the cylinder are clearly visible.	91

7.7. Normal stress τ_{xx} at the symmetry line and on the cylinder surface in the interval $[-1\dots1]$. For larger $Wi > 0.8$ no steady solution could be accomplished.	92
7.8. Pressure distribution on the cusp of the cylinder. For $Wi > 0.8$ no convergent steady solution could be accomplished. It can be seen that the angle between the cross-stream and the stream-wise pressure derivative becomes larger for higher Weissenberg numbers.	92
7.9. Distribution of the velocity u_y on the cusp of the cylinder. For larger $Wi > 0.8$ no convergent steady solution could be accomplished. It can be seen in the detail plot that there is a kink in the distribution close to the cylinder wall. . .	93
7.10. Distribution of the velocity u on the cusp of the cylinder. For larger $Wi > 0.8$ no convergent steady solution could be accomplished. On the detail plots it can be seen that the maximum velocity increases for increasing Weissenberg numbers and the velocity decreases at the cusp of the cylinder.	93
7.11. Evolution in time of the drag force of the confined cylinder for different Weissenberg numbers for unsteady calculations. For higher $Wi \geq 0.6$ there is in the detail plot an oscillatory behaviour which might be caused by unsteady effects due to velocity inflection in the boundary layer.	94
7.12. Drag coefficient depending on the Weissenberg number for different Reynolds numbers.	95
7.13. Distribution of the velocity u_y on the cusp of the cylinder for $Wi = 0.6$ and Newtonian flow for different Reynolds numbers.	95
7.14. Normal stress τ_{xx} at the symmetry line and on the cylinder surface in the interval $[-1\dots1]$ for $Wi = 0.6$ for different Reynolds numbers.	96
7.15. Computational domain of a droplet in shear flow with moving walls. The measurements and BC are filled in.	97
7.16. Mesh of the droplet in shear flow after initial refinement with three refinement levels depending on the curvature of the interface. The interface represented by the zero level-set ($\varphi = 0$) is marked in red.	97
7.17. Drag coefficient depending on the Weissenberg number for different Reynolds numbers.	99
7.18. Contour of the zero level-set at different times of the (NN)-simulation (solid line), the (NV)-simulation (dashed), and the (VN)-simulation (dotted). At $t = 0.075$ the (NV)-simulation had already failed.	99
7.19. Pseudocolour plot of the shear stress τ_{xy} inside and outside of the droplet for the high-order simulation at $t = 0.020$. The stress jump because of the jump in the Weissenberg number can be seen. The arrows are the velocity vectors. . .	100

List of Tables

4.1. Coefficients for the BDF $_z$ schemes for $z \leq 4$ and the Crank-Nicholson scheme (CN). The coefficients in Eq. (4.42) are related to the coefficients in the table as follows: $b_0 := \frac{\nu_0}{\nu}$, $a_j := \frac{\nu_j}{\nu}$ with $1 \leq j \leq 4$. IE is the implicit Euler method which is equal to the BDF1 scheme.	48
7.1. Experimental Order of Convergence (EOC) of Cockburn et al. (2002) (lit.) and of our LDG solver (here) for the Stokes system.	82
7.2. EOC of our LDG solver for the Navier-Stokes system with $Re = 1$ using the manufactured solution in Eq. (7.1).	83
7.3. Amount of nodes of the computational meshes in different regions for the convergence study and corresponding refinement factor for that direction. . .	85
7.4. Number of cells and of degrees of freedom for different mesh sizes and polynomial degrees with k for the velocity and stresses and $k - 1$ for the pressure. . .	86
7.5. Polynomial degree of the discretization and experimental order of convergence (EOC) for both Weissenberg numbers and different dependent variables. . . .	86
7.6. Dimensionless drag force of the confined cylinder for different Weissenberg numbers for steady calculations compared with unsteady calculations from exemplary literature [a] Kim et al. (2004), [b] Claus and Phillips (2013), and [c] Keith et al. (2017).	89
7.7. Dimensionless drag force of the confined cylinder for different Weissenberg numbers for steady calculations for different Reynolds numbers compared with unsteady calculations from exemplary literature [a] Claus and Phillips (2013), and [b] Keith et al. (2017).	94

List of symbols

A	area
A'	Jacobian of operator matrix A
A_M	aggregation map
A_U	operator matrix
\mathcal{A}	arbitrary linear dependent operator matrix
\mathfrak{A}	bulk phase A
\mathbf{a}	acceleration
a	a constant
α	upwinding parameter
α_G	constant in Giesekus model
B	ball around solution in Newton method
\mathbf{B}	left Cauchy-Green or Finger-strain tensor
\mathcal{B}	continuous body
\mathcal{B}_R	continuous body in reference configuration
\mathfrak{B}	bulk phase B
\mathbf{b}	right-hand-side
b_0	constant in BDF
β	dimensionless viscosity, viscosity ratio
C	constant in flux
\mathbf{C}	right Cauchy-Green or Cauchy-Green tensor
Ca	capillary number
\mathbf{c}	conformation tensor
\mathbf{c}	translation
c_s	velocity of propagation
D	deformation parameter of droplet
\mathbf{D}	strain rate tensor
D_v	set of dependent variables
\mathcal{D}	unspecified tensor function space
De	Deborah number
d	dimension
δ	tensor test function
δ	radius of ball around solution in Newton method
$\delta(s)$	Dirac function
δ_{ij}	Kronecker delta

\mathcal{E}	energy functional
\mathbb{E}	Euclidian space
\mathbf{e}	error
\mathbf{e}_i	unit vector in i-th direction
ε	small value
ϵ	factor artificial viscosity
F	force
\mathbf{F}_B	body force
\mathbf{F}	deformation gradient
\mathcal{F}	functional
\mathfrak{F}	Cartesian frame of reference
f	function
f^*	flux
\mathbf{f}	vectorial function
\mathbf{G}	tensor depending on the Finger tensor
\mathbf{g}	auxillary variable in LDG method
Γ	union of all edges of a grid
Γ_I	union of all interior edges of a grid
γ	penalty constant
$\dot{\gamma}$	shear rate
H	Heaviside function
\mathbf{H}	tensor depending on the Cauchy-Green tensor
h	characteristic mesh size
\bar{h}	height
\tilde{h}	height function
η	viscosity
η_0	total viscosity
η_p	polymeric viscosity
η_s	solvent viscosity
\mathbf{I}	unit tensor
\mathfrak{I}	interface between two fluid phases
ι	Lipschitz constant
\mathcal{J}	general Jacobian of nonlinear operator
K	element
\mathcal{K}	Kernel function
\mathfrak{K}	set of geometry-conforming non-overlapping elements
k	total polynomial degree
\mathbf{k}	total polynomial degree
κ	curvature of a surface

\mathbf{L}	velocity gradient
\mathcal{L}	linear operator
LU	LU factorization
l	length
l_c	characteristic length
l_s	surface segment on the interface
Λ	bulk viscosity
λ	eigen value
λ_1	relaxation time
λ_2	retardation time
Δ	Laplace operator
$\underline{\Delta}$	
\underline{S}	lower convected time derivative
\mathbf{M}	two-term Taylor expansion
\mathcal{M}_M	mass matrix
\mathcal{M}_S	stiffness matrix
m	viscosity ratio between fluid phases
μ	security threshold for bounds of smoothness indicator
\mathbf{N}	skew symmetric rotational tensor
N_1	first normal stress difference
N_2	second normal stress difference
\mathbf{n}	normal vector
ν	constants in BDF
∇	Nabla operator
∇_h	broken Nabla operator
\mathcal{O}	Order
\mathbf{P}	projection tensor, the subscripts (\mathcal{I}, S) denotes the surface of projection
\mathbb{P}_k	broken polynomial space of total degree k
p	pressure
pot	potential function
Φ	field
ϕ	polynomial basis
φ	level-set function
Ψ	matrix logarithm
ψ	arbitrary scalar field
\mathbf{Q}	rotation matrix
\mathcal{Q}	unspecified scalar function space
\mathbf{q}	vector test function
q	scalar test function
\tilde{q}	switching variable between BDF and Crank-Nicholson
q_N	quadrature node

R	Residual
Re	Reynolds number
r	radius
res_i	restriction function
ρ	density
S	surface
\mathbf{S}	extra stress tensor
S_h	smoothness indicator
\mathcal{S}	unspecified tensor function space
s	general fluid species
s	time scale
\mathbf{s}	Newton step
s_h	logarithm of smoothness indicator
s_0	critical smoothness indicator value
s_i	sign indicator
Σ	modified polymeric stress
σ	surface tension
σ	tensor test function
T	Cauchy stress tensor
t	time
t_c	characteristic time
\mathbf{t}	surface force
tol	tolerance
τ	polymeric part of the extra stress tensor
Θ	rotation angle
θ	opening angle
\mathbf{U}	solution vector
\mathbb{U}	normed vector space
\mathbf{u}	velocity
u	scalar field
u_c	characteristic velocity
u^*	extensional velocity
∇	
\mathcal{S}	upper convected time derivative
V	volume
\mathcal{V}	unspecified vector function space
\mathbb{V}_k	DG function space for test and trial functions
v	scalar test function
\mathbf{v}	vector test function
\mathbf{W}	vorticity tensor
We	Weber number

Wi	Weissenberg number
w_i	weights
Ω	domain of interest, physical domain
Ω_h	computational domain
$\partial\Omega$	boundary of domain of interest, of physical domain
$\delta\Omega_h$	boundary of computational domain
ω_c	characteristic frequency
\mathbf{X}	position in the reference configuration
x	x coordinate
\mathbf{x}	current position
χ	momentum transfer
ξ	Lagrange multiplier
y	y coordinate
z	order of BDF
\mathbf{z}	tangential vector to the interface
ζ	stress splitting parameter

List of Abbreviations

AVSS Adaptive Viscoelastic Stress Splitting

BC Boundary Condition

BDF Backward Differentiation Formula

BLAS Basic Linear Algebra Subprograms

BoSSS Bounded Support Spectral Solver

CFL Courant-Friedrichs-Lewy

DEVSS Discrete Elastic Viscous Stress Splitting

DFG German Research Foundation (Deutsche Forschungsgemeinschaft)

DG Discontinuous Galerkin

DOF Degrees of Freedom

EEME Explicitly Elliptic Momentum Equation

EOC Experimental Order of Convergence

EVSS Elastic Viscous Stress Splitting

FD Finite Differences

FEM Finite Element Method

FV Finite Volumes

FWF Austrian Science Fund (Fonds zur Förderung der wissenschaftlichen Forschung)

GMRES Generalized Minimal Residual

HDG Hybridizable Discontinuous Galerkin

HPC High Performance Computing

HWNP High Weissenberg Number Problem

IC Initial Condition

LAPACK Linear Algebra Package

LBB Ladyzenskaja-Babuska-Brezzi

LCM Lower Convected Maxwell Model

LDG Local Discontinuous Galerkin

MOL Method of Lines

MPI Message Passing Interface

PDE Partial Differential Equation

PTT Phan-Thien Tanner

SIP Symmetric Interior Penalty

SU Streamline Upwinding

SUPG Streamline Upwinding Petrov-Galerkin

UCM Upper Convected Maxwell Model

XDG eXtended Discontinuous Galerkin

XFEM eXtended Finite Element Method

1. Introduction

In many flow processes the behaviour of the fluid is non-Newtonian. Especially in biological flow and environmental or chemical processes, such as blood flow (Thurston, 1972; Bodnár et al., 2011), sedimentation processes (Connolly and Podladchikov, 2000) or polymer melts (Hu and Granick, 1992), we can observe viscoelastic flow phenomena like the characteristic fading memory effect. In polymer melts and biological flows the viscoelastic behaviour mostly results from untangling long chain molecules where in case of biological flows these are usually amino acids. Viscoelastic behaviour occurring in sedimentation processes arise from the high density particle flow. A dumbbell-shaped structure of the molecules or particles can also increase viscoelastic behaviour.

In these areas of interest, processes such as spraying or the flow of emulsions and polymer melts can be found, leading to droplets displaying viscoelastic behaviour. Examples are spray drying, blood flows, injection moulding, food processing or the production of cosmetics and toiletries. Therefore, the physical behaviour of viscoelastic droplets, and in particular, their oscillatory acting is of great interest in many different areas of application.

The underlying scientific work of this PhD thesis is a project funded by the German Research Foundation (Deutsche Forschungsgemeinschaft) (DFG) and Austrian Science Fund (Fonds zur Förderung der wissenschaftlichen Forschung) (FWF) in cooperation with the Institute of Fluid Mechanics and Heat Transfer, Graz University of Technology, Austria. The aim of this project is a systematic study of the physical behaviour of oscillating viscoelastic droplet. Therefore, the members of the working group of Prof. Brenn (TU Graz) examine the oscillatory behaviour experimentally and by using asymptotic analytical approaches. The aim of the numerical working group at the Chair of Fluid Dynamics is the numerical investigation of the oscillatory behaviour of a viscoelastic droplet using highly accurate computational methods. In the end of the project phase the results obtained with different methods shall be compared and physical properties shall be derived in future.

The aims of this work are to develop and verify a new high-order discretization for the governing equations modelling viscoelastic behaviour based on the DG method and in a second step to extend this discretization for the multi-phase purpose using the XDG method. All this developmental work is embedded in the in-house DG solver framework BoSSS at the chair of fluid dynamics. Up to now, the framework is capable of simulating only Newtonian single- and multi-phase flows using a sharp interface approach with a level-set function.

The outline of this PhD thesis is as follows: After a short introduction into the field of research the reader finds in Chap. 2 the renowned continuum mechanical deduction of the governing equations and some aspects about the modelling of viscoelastic materials. In Chap. 3, the problems arising in the numerical discretization of the viscoelastic system of equations are illuminated, and the standard methods used in the field of viscoelastic flow are presented. In

the following (Chap.4 and 5), the DG discretization for the single-phase case and the multi-phase purpose (eXtended Discontinuous Galerkin, XDG) using the evolution of a level-set function representing the interface are displayed. After a chapter about solving strategies particularly applying to viscoelastic flow (Chap. 6), the reader will find numerical results conducted with the presented solvers (Chap. 7), followed by a short conclusion in the end (Chap. 8).

1.1. Physical Properties of Viscoelastic Material

First of all, the behaviour of fluids is dependent on the ratio of shear stress τ_{xy} to shear rate $\dot{\gamma}$. This ratio is called viscosity and can be seen for many fluids such as water or air as a constant without taking temperature-dependency into account (isothermal). They show a featureless microstructure and are called Newtonian.

As displayed in Fig. 1.1, a fluid can also have a non-linear dependency on the shear rate. If the viscosity is a decreasing function of the shear rate, it is called a shear-thinning fluid. Examples are fluid with long chain microstructures such as polymer melts. The opposite behaviour with the viscosity being an increasing function of the shear rate is called shear-thickening or dilatant which is often due to the formation of clusters in the fluid and can be found in concentrated suspensions and sedimentation flows (Phan-Thien and Mai-Duy, 2017).

If we also take the time dependent behaviour of the viscosity into account, we can distinguish between thixotropic fluids with a time dependent shear-thinning characteristic. A very popular example is ketchup where stick-shaped particles align with continuing shear. The opposite, namely time dependent shear-thickening or rheopectic behaviour, is not very common and can be found in some lubricants.

Whereas in Newtonian fluids the normal stresses τ_{xx} , τ_{yy} and τ_{zz} are equal, there are differences for viscoelastic fluids such that we can define a first and second normal stress difference $N_1 = \tau_{xx} - \tau_{yy}$ and $N_2 = \tau_{yy} - \tau_{zz}$ (Phan-Thien and Mai-Duy, 2017). Note that in this work we consider only two-dimensional flow since the three-dimensional case enlarges the numerical system to be computed, so N_2 is undefined. These normal stress differences lead to a series of counter-intuitive behaviours, some of them are displayed in Fig. 1.2.

The first is the so-called Weissenberg rod climbing effect where the viscoelastic fluid climbs a rod rotating in the fluid (Fig. 1.2a). The reason is the fluid elements being able to support a tension along their streamlines due to the non-zero normal stress differences. This leads to a tension force pointing upwards the rod. The same effect plays a role in the open siphon experiment where a viscoelastic fluid does not stop to flow out a beaker against gravity (Fig. 1.2c). The next effect is the die swell which is significantly larger for viscoelastic than for Newtonian fluids (Fig. 1.2b). The first normal stress lets the viscoelastic fluid swell after emerging from a capillary when entering the free surface flow. For increasing flow velocities, inertia tends to reduce the amount of swell and delay the point of swelling. In case of a complete free surface flow, when letting a viscoelastic fluid flow down an inclined plate, it shows a convex shape because of the existence of a negative second normal stress difference although it is small. In case of a Newtonian flow the free surface is nearly flat and only influenced by the surface tension. The existence of the normal stress differences also leads to the reversal of many secondary flow patterns such as Taylor vortices (Phan-Thien and Mai-Duy, 2017).

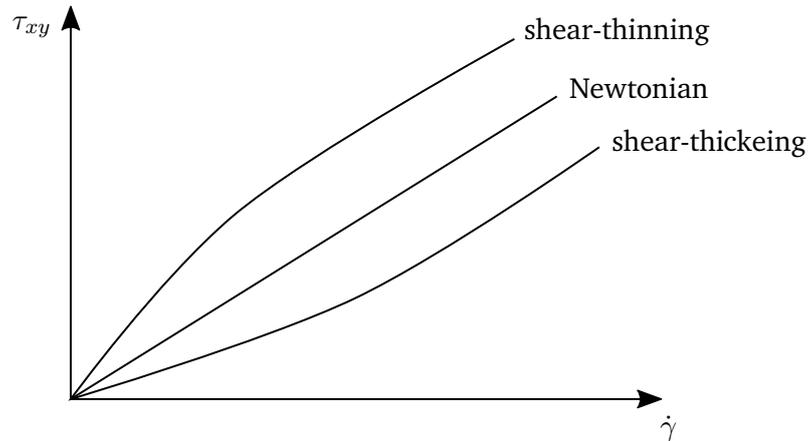


Figure 1.1.: Dependency of the shear stress on the shear rate. The function displays the viscosity of the fluid for Newtonian, shear-thinning and shear-thickening (dilatant) behaviour.

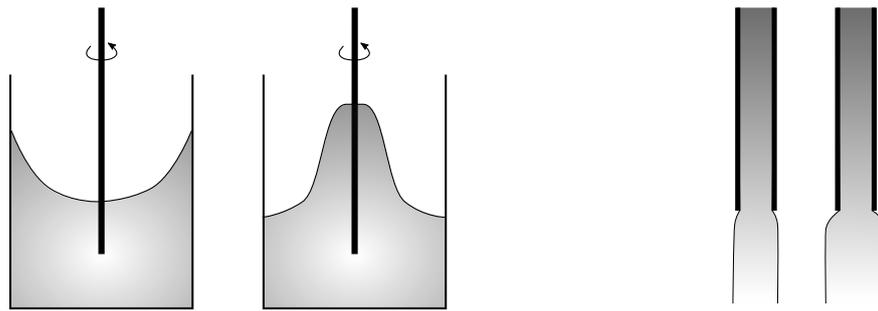
Another effect in viscoelastic flow is the fading memory over time. While Newtonian fluids have almost no memory by the motion of the fluid ceasing immediately in the moment where the loading is removed, elastic solids have a perfect memory by returning to their original shape after the loading stops. The behaviour of a viscoelastic fluid lies in between, partially returning to a stage before the moment when loading is stopped (Phan-Thien and Mai-Duy, 2017). One example of a fading memory is recoiling where the viscoelastic fluid, which is cut off while pouring down, partially retracts back into a beaker (Fig. 1.2d).

The behaviour of the viscoelastic fluid is quantified by its stress relaxation and strain retardation time. The stress relaxation is the reaction of the stress of the fluid over time after a loading with a constant shear rate occurs, the strain retardation is the delayed deformation or shear rate responding on a constant stress (Fig. 1.3).

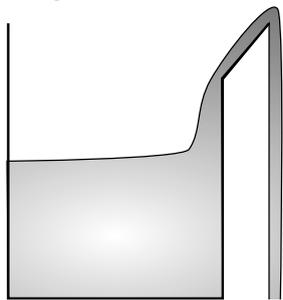
Because of the non-linear characteristic of viscoelastic fluids, these phenomena lead to a lot of physical instabilities which can be observed in experiments and industrial applications. These instabilities are mainly driven by the normal stress differences in combination with the nature of some BC. Severe instabilities can occur in Taylor-Couette flows, torsional flows between two disks, curved pipes, contractions or extrusion tools (Phan-Thien and Mai-Duy, 2017). The last one is a classical problem in industrial applications and called melt fracture. It is due to an instability caused by the interaction of the viscoelastic fluid and the nature of the BC (e.g. Ketata et al., 2017; Ebrahimi et al., 2018).

1.2. Physical Properties of Droplets

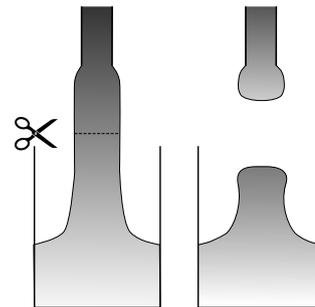
In case of two-phase flow we have an interface between two immiscible liquids or a liquid and a gas. This interface is deformable which means it can change its shape freely trying to minimize its surface energy. In case of a droplet, e.g. a drop of oil in water or a soap bubble, the state of minimal surface energy is a perfect sphere which is smooth on an atomic scale and which is hardly deformable. The typical shape of drops, e.g. of a drop of water falling down



(a) Weissenberg rod climbing effect. Left: Newtonian Fluid, right: viscoelastic fluid. (b) Die swell effect. Left: Newtonian Fluid, right: viscoelastic fluid.



(c) Open siphon effect.



(d) Recoiling effect.

Figure 1.2.: Different effects in viscoelastic fluids due to normal stress differences and a fading memory.

in air is a compromise between the effect of minimizing its surface energy which favours a sphere and gravity or any other force field which causes distortions in shape (de Gennes et al., 2004).

The physical origin of the so-called surface tension can be found in the atomic state of a liquid forming a droplet. A molecule in the middle of a liquid has interactions with all its neighbours while a molecule that wanders to the surface of the droplet loses half of its cohesive interactions (Fig. 1.4). This is the fundamental reason why the liquid adjusts its shape in order to expose the smallest possible surface area (de Gennes et al., 2004). If we consider the cohesion energy of a molecule inside a droplet, then a molecule at the surface loses roughly half of this energy. The surface tension σ can be viewed as a direct measure of this shortfall of energy per unit on the surface. Note that although we explain the origin of the surface tension on an atomic level, the surface tension itself is a parameter defined on a macroscopic scale. This energy point of view can also be clarified by considering that it is necessary to supply energy in order to create or increase the surfaces of a liquid, e.g. by beating egg whites to produce a foam or by blowing soap bubbles. If we want to distort the liquid to increase its surface area by dA , the work which is required is proportional to the amount of molecules that must be brought up the surface by $dW = \sigma \cdot dA$. So the surface tension σ is the force per unit that must be supplied to increase the surface area by one unit (de Gennes et al., 2004). Another phenomenon of droplets is that the surface tension provides an overpressure inside the drop or bubble. The arising pressure difference between outside and inside the drop can be explained by a drop of oil in water. In order to lower its surface energy, the drop takes a spherical shape of radius R . The oil is denoted with a subscript O and the water with a

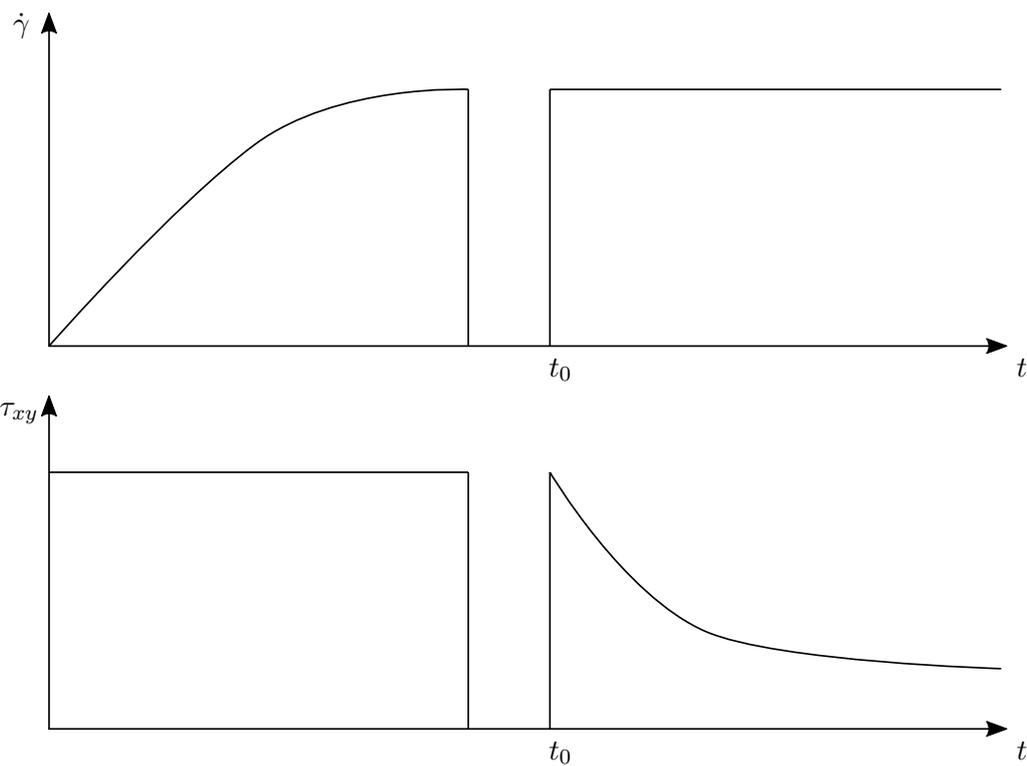


Figure 1.3.: Strain retardation and stress relaxation. On the left side a constant stress is applied to the viscoelastic fluid in the lower graph. The strain response is shown in the upper graph displaying a retardant behaviour slowly saturating to a final shear rate. At time t_0 a constant shear rate is applied to the fluid causing a relaxing stress response until the final stress is reached.

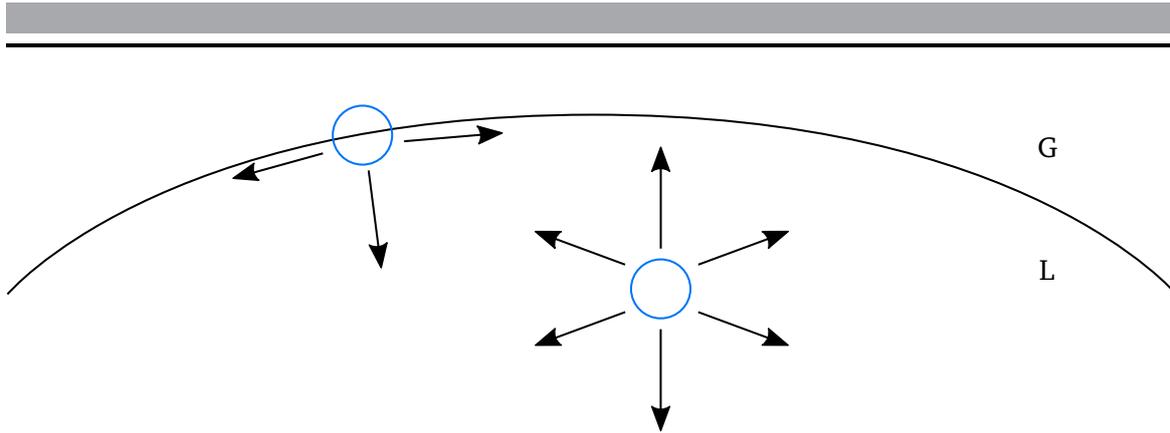


Figure 1.4.: Schematic representation of a molecule within a liquid drop in a gaseous ambient medium and one molecule at the surface of the drop. The arrows indicate the cohesion energy with the surrounding molecules. It can be seen that this energy is roughly halved at the surface leading to a tension at the surface.

subscript W . The work is then $dW = p_O dV_O - p_W dV_W + \sigma dA$. If we now consider the volumes as $dV_O = 4\pi r^2 dr = -dV_W$, and the surface area as $dA = 8\pi r dr$ in dependency of the drop radius and introduce a mechanical equilibrium with $dW = 0$, we gain a relation between the pressure jump and the drop radius with $\Delta p = p_O - p_W = \frac{2\sigma}{r}$. As we can see, the smaller the drop the greater is the inner pressure. Experimentally, this can be observed by connecting two bubbles or droplets of different size, since the smaller drop empties itself into the larger drop because of the higher pressure inside (de Gennes et al., 2004).

1.3. Motivation and Objectives of this Work

The simulation of fluids with viscoelastic behaviour is particularly challenging due to two major issues: First, the proper modelling of the physical characteristics or the right choice of a viscoelastic model is crucial in order not to develop unphysical models or even ill-posed problems with non-unique or even inexistent solutions (Owens and Phillips, 2002). Existing models such as the Oldroyd B model used in this work are convection-dominated with a missing diffusion term in the constitutive equations. Thus, they are of hyperbolic nature (Joseph et al., 1987). Second, the numerical method has to deal with stability and accuracy problems caused by a strong variation of length scales within the thin boundary layer, where velocity gradients and stresses can rapidly change their values by several orders of magnitude (Dou and Phan-Thien, 2007). Both leads to a loss of convergence even for minimal elastic behaviour in the fluid.

A good approach is the use of the DG method, because its discontinuous elements with appropriate flux functions for the edges make it more robust against numerical oscillations compared to e.g. Continuous Galerkin methods. Within the last 25 years, the DG method has been successfully established for solving hyperbolic conservation laws and was first introduced by Fortin and Fortin (1989) for viscoelastic fluid flow. It is also strongly emerging in other fields of computational fluid dynamics (Cockburn et al., 2002). There are two reasons for this ascent which obviates essential limitations of classical techniques such as Finite Volumes (FV) or Finite Differences (FD) methods. DG cleverly combines an arbitrary order $k \in \mathbb{N}$ in

the numerical discretization error $\mathcal{O}(h^{k+1})$ with a local flux evaluation which is at most to be computed from adjacent cells. Here, h refers to the local grid spacing, and k to the order of the DG basis polynomials. This is in strong contrast to the established schemes such as FV methods, which are substantially limited to a convergence order of two on unstructured grids, and even on Cartesian grids it is rather limited to small values because of the larger stencils required for increasing convergence order (Cockburn et al., 2000).

In viscoelastic flow the DG method is often used to obtain stability for the convection-dominated convection-diffusion problem using a streamline upwinding formulation. In this case, the convective term of the constitutive equations is discretized by a DG scheme whereas the other variables in the momentum and continuum equations are discretized using a standard Finite Element Method (FEM). The DG method allows jumps in the BC and preconditioning at the elemental level, appropriate flux functions for the edges can be chosen and velocity-stress compatibility conditions can be easily satisfied (Owens and Phillips, 2002). In conclusion, the DG method is a promising method for convection-dominated problems. However, in the context of viscoelastic flow there are few approaches fully based on this method. First ideas for high order DG in a decoupled scheme can be found in the newer work of Mirzakhilili and Nejat (2015).

A breakdown in convergence can also occur due to the mixed hyperbolic-elliptic-parabolic type of the system of equations. Whereas the saddle-point problem of the Navier-Stokes system is of elliptic type, the constitutive equations modelling the viscoelastic behaviour are hyperbolic. As it is shown in Chap. 3, the viscous part of the momentum equations and of the constitutive equations are weighted by a material parameter β . If β is close to 1, the contribution of the constitutive equations is small and we have to solve an elliptical system. If $\beta \rightarrow 0$ such that we have a highly elastic fluid for increasing Weissenberg numbers without retardation effects, a change of type from elliptic to hyperbolic occurs and the numerical solution becomes unstable unless special care is taken (Joseph and Saut, 1986; Joseph et al., 1987; Joseph, 1990). There are several approaches in numerical computation for handling the strong mixed hyperbolic-elliptic coupling between the momentum and constitutive equations by the velocity gradient. In the Elastic Viscous Stress Splitting (EVSS) method and its derivatives a second-order elliptic term is introduced in the constitutive equations and the depending variables are changed such that there is no necessity for additional compatibility conditions for the well-posedness of the discrete system in the Stokesian limit (Owens and Phillips, 2002; Kim et al., 2004; Sun et al., 1999; Dou and Phan-Thien, 2007). However, this extends the system of equations to be solved by an additional evolutionary equation for the velocity gradient.

We aim for a solver for viscoelastic flow with an exclusively high-order DG scheme for all equations using a LDG formulation with penalized fluxes in order to solve the hyperbolic constitutive equations and using a streamline upwinding for the convective fluxes of the constitutive equations. The solver is embedded in the open source DG framework BoSSS, currently under development at the Chair of Fluid Dynamics of Technical University of Darmstadt, which can be downloaded under <https://github.com/FDYdarmstadt/BoSSS>.

The BoSSS framework was initiated in 2008 in order to establish a foundation for the development of high-order discretization for challenging physical problems (Kummer et al., 2020a). Up to now it evolved into a fully-featured library for DG methods containing not only the numerical discretization for many use cases but also facilities for a workflow management and the rapid prototyping discretization for different Partial Differential Equation (PDE). The idea behind the BoSSS code is to provide a research code environment bridging the gap

between prototypes with limited performance but huge generality (such as MATLAB) and highly optimized single-purpose research codes needing advanced skills from the user on the other side. This is also the reason why the software is written in the programming language C# which is a strong easy-to-use language comparable with Python whereas the execution speed is comparable with C or C++. It can also be compiled once and executed everywhere, e.g. on a high-performance Linux-based supercomputer without the need of special configuration (Kummer et al., 2020a).

The structure of the BoSSS framework is illustrated in Fig. 1.5. The end user can access different upper level applications for different use cases such as multi-phase flows e.g. using a BoSSSpad worksheet similar to that of prototype programs such as MATLAB. In the BoSSS herd of libraries, many different accessible methods can be found such as the implementation of the DG discretization of different partial differential terms, time discretization schemes or non-linear solvers. For these utilities BoSSS uses different third party libraries like Message Passing Interface (MPI) for parallel computing or linear algebra tools such as Basic Linear Algebra Subprograms (BLAS) or Linear Algebra Package (LAPACK).

In this work the DG discretization of the governing equations for a viscoelastic Oldroyd B fluid and some methods presented in Chap. 6 were implemented and embedded in the herd of libraries. Two solvers within the application layer were written: the Rheology_Solver for applications of single-phase viscoelastic flow and the XRheology_Solver for the multi-phase purpose. The structure of all methods implemented is kept general in a term-by-term wise manner such that other non-Newtonian (viscoelastic or non-viscoelastic) fluids can be easily supplemented introducing additional terms. Since the LDG method was implemented adding a stress tensor to the dependent variables, it can also be useful in a broader range of applications such as turbulence models or other tensor-valued quantities.

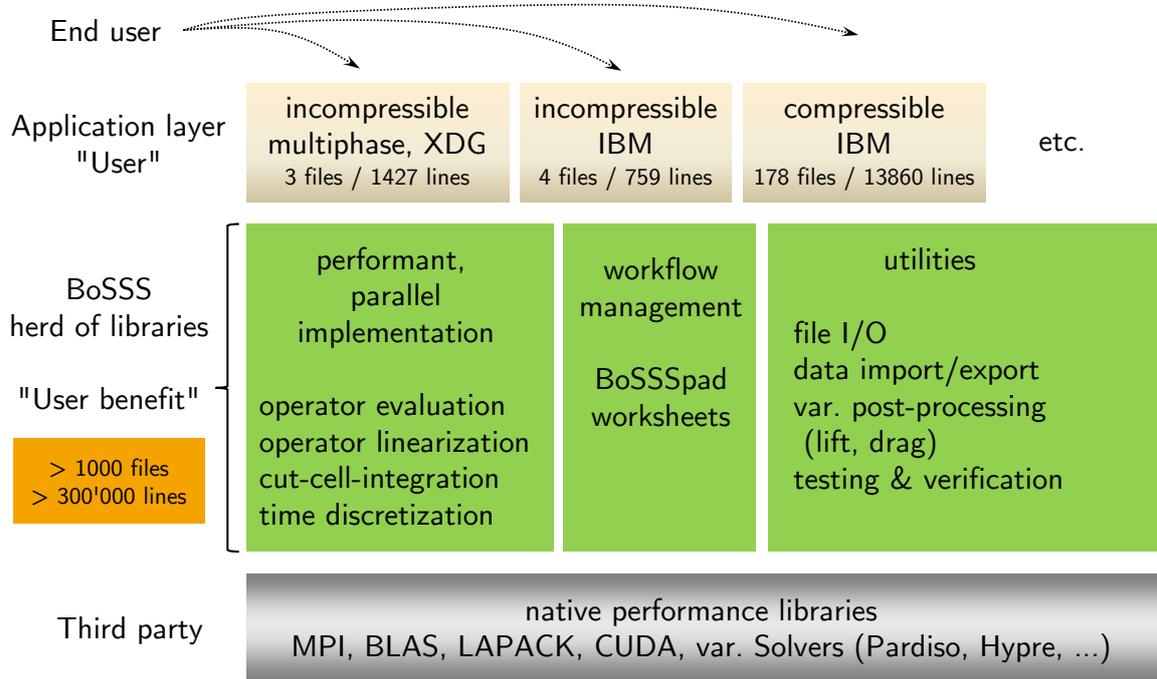


Figure 1.5.: Schematic representation of the structure of the BoSSS solver from the BoSSS handbook (Kummer et al., 2020a).

2. Governing Equations

In this chapter we introduce the governing system of equations consisting of the continuity equation for incompressible flow with $\rho = \text{const.}$, the momentum equations and the constitutive equations describing the viscoelastic behaviour of the material. To understand the mathematical description of the physics of viscoelastic flow and instability issues in the numerical simulation treated in Chap. 3, a full deduction of the equations including the procedure of rheological modelling is presented.

The contents in this chapter are mainly deduced from the books of Phan-Thien and Mai-Duy (2017) and of Giesekus (1994). For a deeper insight into the continuum mechanics we refer the reader to the previously mentioned sources.

First, some continuum mechanical definitions and correlations used in the outline of this thesis are presented. Afterwards, the continuity and momentum equations are derived. Then some insight into the rheological modelling is given, and in the end, the governing system of equations for the dimensional and dimensionless case for single-phase and multi-phase flow are presented.

2.1. Continuum Mechanical Definitions

Since we only consider two-dimensional flow, we have a two-dimensional Euclidean space \mathbb{E} containing a set of vectors and tensors building a normed vector space \mathbb{U} and a Cartesian frame of reference $\mathfrak{F} = \{0; \mathbf{e}_{i1}, \mathbf{e}_{i2}\}$ with an origin 0 and an orthonormal basis $\{\mathbf{e}_{i1}, \mathbf{e}_{i2}\}$.

All elements in the vector space \mathbb{U} must be invariant under a change of frame, especially under frame rotation with a rotation angle Θ :

$$\cos(\Theta) = Q_{ij} \mathbf{e}_{ii} \cdot \mathbf{e}_{ij}, \quad (2.1)$$

where the orthogonal Matrix Q is the rotation matrix.

We have a continuous body \mathcal{B} occupying a region which consists of points in \mathbb{E} . This body in the reference configuration is denoted as \mathcal{B}_R , e.g. at time $t = 0$. To describe the motion, velocity and acceleration of particles within the body \mathcal{B} we use an Eulerian description referring to these fields as functions of the current position \mathbf{x} . Using $\mathbf{u}(\mathbf{x}, t)$ as the Eulerian velocity field, the acceleration \mathbf{a} reads:

$$\mathbf{a} := \frac{d}{dt} \mathbf{u}(\mathbf{x}, t)|_{\mathbf{x}} = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \quad (2.2)$$

where \mathbf{X} is the position of the particle in the reference configuration. We call this derivation, consisting of a time derivative and a convection term, the material derivative or total time derivative.

To describe the local deformation of a material point, e.g. a fluid element, with respect to the reference configuration we use the deformation gradient:

$$\mathbf{F} := \frac{\partial \mathbf{x}}{\partial \mathbf{X}} \quad (2.3)$$

such that:

$$d\mathbf{x} = \mathbf{F}d\mathbf{X} \quad d\mathbf{X} = \mathbf{F}^{-1}d\mathbf{x} \quad (2.4)$$

with $d\mathbf{x}$ is a fluid element at its current position \mathbf{x} , whereas $d\mathbf{X}$ is the fluid element at the reference position \mathbf{X} . Using the deformation gradient we can define the right Cauchy-Green or Cauchy-Green tensor:

$$\mathbf{C} := \mathbf{F}^T \mathbf{F}, \quad (2.5)$$

and the left Cauchy-Green or Finger-strain tensor

$$\mathbf{B} := \mathbf{F} \mathbf{F}^T, \quad (2.6)$$

such that $\mathbf{B} = \mathbf{C}^{-1}$. Both symmetric tensors are always positive definite since their Eigenvalues and therefore, their determinants are positive.

In the case of the continuum mechanics of fluid flow, the velocity gradient defined by:

$$\mathbf{L} = \nabla \mathbf{u} \quad (2.7)$$

is used to describe material and momentum properties. It can be split in a symmetric and a skew-symmetric part:

$$\mathbf{L} := \mathbf{D} + \mathbf{W}, \quad (2.8)$$

where $\mathbf{D} := \frac{1}{2} (\nabla \mathbf{u} + (\nabla \mathbf{u})^T)$ is the symmetric strain rate tensor and $\mathbf{W} := \frac{1}{2} (\nabla \mathbf{u} - (\nabla \mathbf{u})^T)$ is the skew-symmetric vorticity tensor. This decomposition is important for ensuring objectivity which is further explained in section 2.4.

Further, the Gauss divergence theorem is for a scalar, a vector, and a tensor of second order:

$$\int_V \nabla \psi \, dV = \int_S \psi \mathbf{n} \, dS, \quad \int_V \nabla \cdot \mathbf{u} \, dV = \int_S \mathbf{u} \cdot \mathbf{n} \, dS, \quad \int_V \nabla \cdot \mathbf{T} \, dV = \int_S \mathbf{T} \cdot \mathbf{n} \, dS, \quad (2.9)$$

where \mathbf{n} is the outward pointing normal vector, and the Leibniz formula:

$$\frac{d}{dt} \int_V \psi \, dV = \int_V \frac{\partial \psi}{\partial t} \, dV + \int_S \psi \mathbf{u}_S \cdot \mathbf{n} \, dS \quad (2.10)$$

with \mathbf{u}_S is the changing velocity of the surface $S(t)$. Using these relations (2.9) and (2.10), we can derive the Reynolds transport theorem:

Theorem 2.1 *Let $\Phi(\mathbf{x}, t)$ be a field (scalar-, vector- or tensor-valued) defined over a region V occupied by the body \mathcal{B} at the time t , then the time derivative is:*

$$\frac{d}{dt} \int_V \Phi \, dV = \int_V \left(\frac{d\Phi}{dt} + \Phi \nabla \cdot \mathbf{u} \right) \, dV. \quad (2.11)$$

Theorem 2.2

$$\frac{d}{dt} \int_V \Phi \, d\mathbf{x} = \int_V \frac{\partial \Phi}{\partial t} \, dV + \int_S \Phi (\mathbf{u} \cdot \mathbf{n}) \, dS. \quad (2.12)$$

Here, it can be seen that the quantity Φ changes in time by a rate of creation and a flux into the volume V through its boundary surface S .

2.2. Conservation of Mass

Since the total mass of a defined volume within the body \mathcal{B} may not change, we have to ensure that the change over time for the density within the volume must be conservative:

$$\frac{d}{dt} \int_V \rho \, dV = 0. \quad (2.13)$$

Using the Reynolds transport theorem as in the form of (2.11) we obtain:

$$\int_V \left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \right) \, dV = 0, \quad (2.14)$$

and since the chosen volume is arbitrary, the necessary and sufficient condition reads:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0. \quad (2.15)$$

For an incompressible and isothermal fluid with $\rho = \text{const.}$ the equation (2.15) simplifies to:

$$\nabla \cdot \mathbf{u} = 0. \quad (2.16)$$

2.3. Conservation of Momentum

If we consider Newton's law of linear momentum, we can derive the conservation of momentum. Thus, the sum of all forces acting on the body \mathcal{B} consists of body forces such as gravity and surface traction forces. The resulting force due to body force is defined by

$$\int_V \rho \mathbf{F}_B \, dV. \quad (2.17)$$

Consider a particle \mathbf{X} in the current position \mathbf{x} at time t , the surface force $\mathbf{t}(\mathbf{x}, t; \mathbf{n})$ is the force with which the material is acting on an infinitesimal surface ΔS with a normal vector $\mathbf{n}(\mathbf{x}, t)$. Here, the force $\mathbf{t}(\mathbf{x}, t; \mathbf{n})$ on the positive side of the surface defined by $\mathbf{n}(\mathbf{x}, t)$ is pulling on the negative side of the surface. The resulting surface force is defined by:

$$\int_S \mathbf{t} \, dS. \quad (2.18)$$

Using the forces (2.17) and (2.18) the linear momentum is

$$\frac{d}{dt} \int_V \rho \mathbf{u} \, dV = \int_S \mathbf{t} \, dS + \int_V \rho \mathbf{F}_B \, dV. \quad (2.19)$$

Since the surface traction force satisfies $\mathbf{t}(\mathbf{x}, t; -\mathbf{n}) = -\mathbf{t}(\mathbf{x}, t; \mathbf{n})$, we can introduce a second order tensor field:

$$\mathbf{t}(\mathbf{x}, t; \mathbf{n}) = \mathbf{T}(\mathbf{x}, t) \cdot \mathbf{n}, \quad (2.20)$$

which is called the Cauchy stress tensor and which is material dependent. In this tensor \mathbf{T} the first index of the components T_{ij} marks the surface on which the force is acting whereas

the second index is the direction of the force component. In Sec. 2.4 we go into detail about the characteristics of the tensor and its modelling. Inserting relation (2.20) into (2.19) and transforming the surface integral to a volume integral using the Gauss divergence theorem (2.9), we obtain:

$$\int_V \rho \frac{d\mathbf{u}}{dt} dV = \int_V \nabla \cdot \mathbf{T} dV + \int_V \rho \mathbf{F}_B dV. \quad (2.21)$$

Since the integrand is continuous on an arbitrary volume V , the conservation constraint is sufficient:

$$\rho \frac{d\mathbf{u}}{dt} = \nabla \cdot \mathbf{T} + \rho \mathbf{F}_B. \quad (2.22)$$

This equation is called Cauchy's equation of motion.

2.4. Rheological Modelling

In this section, we recall constitutive equations for the Cauchy stress tensor which describe the material properties of different fluids. We start with the simplest models extending the constitutive equations until they display viscoelastic behaviour. The constitutive models are important to close the under-determined system of equations with three balance equations consisting of the conservation of mass and momentum (Sec. 2.2 and 2.3), but six dependent variables for the two-dimensional case $(\mathbf{u}, p, \mathbf{T})$. Within the dependent variables, there are only three components of the stress tensor. This is because it is symmetric as can be seen in the conservation equation for angular momentum which is not deduced in this work (e.g. in Phan-Thien and Mai-Duy, 2017). We simplify the material modelling by considering only isothermal problems such that we do not have temperature-dependent behaviour in the material.

2.4.1. Inviscid and Newtonian Fluids

We can decompose the Cauchy stress into an isotropic and a deviatoric part \mathbf{T}^D :

$$\mathbf{T} = -p\mathbf{I} + \mathbf{T}^D. \quad (2.23)$$

We start with the most simple model for the Cauchy stress for inviscid flow where the deviatoric part of the stress is zero:

$$\mathbf{T} = -p\mathbf{I} \quad (2.24)$$

where p is the pressure in the fluid. Introducing the model into the momentum equation (2.22) leads to the Eulerian equations for inviscid flow.

If the model (2.24) is extended by an isotropic tensor function of the strain rate tensor \mathbf{D} , we obtain a viscous fluid with a frictional term. By introducing a linear viscosity, we obtain a Newtonian behaviour:

$$\mathbf{T} = -p\mathbf{I} + 2\eta\mathbf{D} \quad (2.25)$$

with \mathbf{D} is the strain rate tensor. The viscosity η is defined by the relation:

$$\eta = \frac{\mathbf{T}_{xy}}{\dot{\gamma}}, \quad \dot{\gamma} = \frac{du_x}{dy} + \frac{du_y}{dx} \quad (2.26)$$

with T_{xy} is the shear stress and $\dot{\gamma}$ is the shear rate, e.g. for a Couette flow the characteristic quantities for the calculation of η are the wall shear stress and the gradient of the wall velocity of the moving wall to the fluid at rest.

In the case of a Newtonian flow, the viscosity is constant. If η is a decreasing function of the shear rate $\dot{\gamma}$, we call it a shear-thinning behaviour, which can be found in fluids with long-chained molecules such as polymer melts and solutions or biological fluids with amino acids. If η is an increasing function of $\dot{\gamma}$, the fluid is shear-thickening. These are mostly suspensions with particles that form clusters while shearing.

It should be noted that the correct formulation for a Newtonian fluid considering a possible volumetric change is the following:

$$\mathbf{T} = -p\mathbf{I} + \Lambda \text{tr}(\mathbf{D})\mathbf{I} + 2\eta\mathbf{D} \quad (2.27)$$

where Λ is the bulk viscosity. But since the pure volumetric change does not affect the value of the stresses, $\text{tr}\mathbf{T}$ is independent of $\text{tr}(\mathbf{D})$. Furthermore, we can absorb the term $\Lambda \text{tr}(\mathbf{D})\delta_{ij}$ into the pressure term. For an incompressible fluid with $\rho = \text{const.}$ the identity is $\text{tr}(\mathbf{D}) = \nabla \cdot \mathbf{u} = 0$.

Inserting the relation (2.25) into the momentum equation (2.22) leads to the well-known Navier-Stokes equations for Newtonian flow which can be used in a wide range of flow problems, especially for water-like fluids.

2.4.2. Principles in Rheological Modelling

Two distinct approaches exist to display the behaviour of the fluid. The first is the continuum or macro-rheological approach and the second the microstructural approach. In the first, the material is viewed as a continuum with no micro-inertial properties. The dependent variables such as the stress are invariant under change of frame and display the rheological behaviour as simple as possible, resulting in general equations containing functions or functionals which have to be determined by experiments with the corresponding material. The advantage for developing a numerical solver is the wide range of applications in using one model describing the rheological behaviour. Simultaneously, these models suffer from a missing accuracy when displaying the specific behaviour of one material.

In the second approach a physical model, which represents the microstructure of the material by using physical principles like Newton's law or conservation constraints, is used to determine the average stresses of the material. These models are in general more specialized to the relevant material and thus not applicable for a wide range of materials.

In this work, with no specification of a certain material, we focus on the continuum approach with the aim to implement a simple model which can be used for a wide range of simulations with different viscoelastic materials without the need to change the terms to be discretized. Especially, when following the continuum approach, there are some principles firstly stated by Oldroyd and Wilson (1950) and later formalized by Noll (1958), which have to be taken into account:

Principle 1 *Material Objectivity.* A constitutive operator is objective or frame-invariant, if it is the same for all observers in relative motion and does not change under a transformation such as translation and rotation.

To explain this principle we concentrate on the case where the change of frame consists of a spatial translation by $\mathbf{c}(t)$ and a rotation by an orthogonal tensor $\mathbf{Q}(t)$ (Eq. (2.1)):

$$\mathbf{x}' = \mathbf{c}(t) + \mathbf{Q}(t) \mathbf{x}. \quad (2.28)$$

A scalar field is always invariant ($\psi' = \psi$), whereas a vector must obey the transformation $\mathbf{u}' = \mathbf{Q}(t) \mathbf{u}$ and a tensor of second order is invariant under $\mathbf{T}' = \mathbf{Q}(t) \mathbf{T} \mathbf{Q}(t)^T$, so only the rotational matrix \mathbf{Q} is involved in the invariant transformation. This principle ensures that for every change of reference the same constitutive equation can be taken into account, or in other words, that the constitutive operator remains the same for every observer in relative motion to the considered body \mathcal{B} .

We need to consider that the velocity gradient is not objective under a change of reference since the Eulerian description of the velocity is dependent on the position of the particle in the reference configuration \mathcal{B}_R . Thus, the velocity transforms:

$$\mathbf{u}'(\mathbf{x}, t) = \dot{\mathbf{c}}(t) + \mathbf{Q}(t) \cdot \mathbf{u}(\mathbf{x}, t) + \dot{\mathbf{Q}}(t) \mathbf{x} \quad (2.29)$$

and the velocity gradient transforms:

$$\begin{aligned} L'_{ij} &= \frac{\partial u'_i}{\partial x'_j} = \frac{\partial u'_i}{\partial x_k} \frac{\partial x_k}{\partial x'_j} = \left(Q_{il} \frac{\partial u_l}{\partial x_k} + \dot{Q}_{ik} \right) Q_{kj}^T, \\ L' &= \mathbf{Q} \cdot \mathbf{L} \cdot \mathbf{Q}^T + \dot{\mathbf{Q}} \cdot \mathbf{Q}^T. \end{aligned} \quad (2.30)$$

Since $\mathbf{Q} \cdot \mathbf{Q}^T = \mathbf{I}$, the part, which violates the objectivity of the velocity gradient \mathbf{L} , is skew-symmetric:

$$\dot{\mathbf{Q}} \cdot \mathbf{Q}^T + \mathbf{Q} \cdot \dot{\mathbf{Q}}^T = 0, \quad \dot{\mathbf{Q}} \cdot \mathbf{Q}^T = -\mathbf{Q} \cdot \dot{\mathbf{Q}}^T = -\left(\dot{\mathbf{Q}} \cdot \mathbf{Q}^T \right)^T. \quad (2.31)$$

This leads to the fact that by decomposition of the velocity gradient (2.8) we obtain an objective symmetric strain rate tensor and a non-objective vorticity tensor:

$$\mathbf{D}' = \mathbf{Q} \cdot \mathbf{D} \cdot \mathbf{Q}^T, \quad \mathbf{W}' = \mathbf{Q} \cdot \mathbf{W} \cdot \mathbf{Q}^T + \dot{\mathbf{Q}} \cdot \mathbf{Q}^T. \quad (2.32)$$

Therefore, the strain rate tensor is preferred describing the dependency of the stress \mathbf{T} on the velocity gradient like in the Newtonian model (2.25).

Principle 2 Local Action. *Only the field directly around the point of interest should be involved in determining the stress.*

Thus, long-range forces should be excluded in the modelling of a material, since they are already included in the body forces which act on the whole body (e.g. gravity).

Principle 3 Determinism. *The current state of the stresses may only be determined by the past history of the motion.*

This means that no future state may be part of the description of the behaviour.

Principle 4 Symmetry. *The symmetry of the stress tensor must always be preserved.*

2.4.3. Normal Stress Differences and Fading Memory

In all the materials described in section 2.4.1, the behaviour is strongly shear dependent and we have equal or zero normal stresses. If the normal stresses are unequal, we can define the first and second normal stress difference:

$$N_1 := T_{11} - T_{22}, \quad N_2 := T_{22} - T_{33}. \quad (2.33)$$

The second normal stress difference N_2 is usually several magnitudes smaller than the first normal stress difference N_1 and does not exist in the two-dimensional case. The normal stress differences are typical for viscoelastic flows and lead to several phenomena only seen in viscoelastic flow such as the rod climbing effect, the die swell, characteristic secondary flow pattern or recoiling. In all these effects a fading memory behaviour contrary to the complete memory of an elastic solid or the complete loss of memory for a Newtonian fluid can be seen. To display the viscoelastic behaviour of a fluid with the fading memory while taking the principles (2) and (3) of modelling into account, we have to define the stress tensor \mathbf{T} as a functional of time and place:

$$\mathbf{T}(\mathbf{x}(\mathbf{X}, t)) := \mathcal{F}_{t'=-\infty}^t \{ \mathbf{x}(\mathbf{X} + d\mathbf{X}, t'), \mathbf{X} \}. \quad (2.34)$$

Now we define that for all times before a time t_0 , where a deformation of the fluid starts, we have a basic incompressible and inviscid state:

$$\mathbf{T}(t) = -p\mathbf{I}, \quad \text{for } t < t_0, \quad (2.35)$$

and the norm of the stress tensor \mathbf{T} , which we define as

$$|\mathbf{T}(t)| := \text{tr}(\mathbf{T} \cdot \mathbf{T})^{\frac{1}{2}}, \quad (2.36)$$

should have the limiting case defined in (2.35) for $t \rightarrow -\infty$, e.g.

$$\lim_{t \rightarrow -\infty} |\mathbf{T}(t)| = \exp^{-at^2}, \quad \text{for } a > 0. \quad (2.37)$$

From such a restriction, we can derive, for example, a linear viscoelastic model.

2.4.4. Viscoelastic Models

If we generalize the idea in section 2.4.3, namely equations (2.34) and (2.37), by taking principle (1) of section 2.4 into account, we obtain a relation of the form

$$\mathbf{Q}^T \cdot (t) \cdot \mathbf{T}(t) \mathbf{Q}(t) = \mathbf{f}(\mathbf{C}(t)) + \mathcal{F}_{s=0}^{\infty} \{ \mathbf{G}_t(t-s); \mathbf{C}(t) \}. \quad (2.38)$$

In this relation the Cauchy-Green tensor and a relative tensor $\mathbf{G}_t := \frac{1}{2}(\mathbf{B}_t - \mathbf{I})$ depending on the relative Finger tensor are used. The index refers to the dependency on the relative deformation gradient \mathbf{F}_t with:

$$\mathbf{F}(\mathbf{x}(t-s)) = \mathbf{F}_t(\mathbf{x}(t-s)) \cdot \mathbf{F}(\mathbf{x}(t)). \quad (2.39)$$

The function $\mathbf{f}(\mathbf{C}(t))$ describes an elastic part with unrestricted memory and the fading memory is constructed by the functional \mathcal{F} such that qualitatively the influence of $\mathbf{G}_t(t-s)$ on the stress tensor $\mathbf{T}(t)$ for each value of $\mathbf{C}(t)$ becomes smaller with increasing distance s to the actual time t .

If we now approximate the functional in Eq. (2.38) using multi-integrals analogously to a Taylor expansion, we obtain for a fluid with constant density using the relation to the Cauchy-Green tensor $\mathbf{H}_t := \frac{1}{2}(\mathbf{I} - \mathbf{C}_t)$:

$$\begin{aligned} \mathbf{T}_A(t) = & -p\mathbf{I} + \int_0^\infty \mathcal{K}_1(s_1)\mathbf{H}_t(t-s_1) \, ds_1 \\ & + \int_0^\infty \int_0^\infty \mathcal{K}_2(s_1, s_2)\mathbf{H}_t(t-s_1) \cdot \mathbf{H}_t(t-s_2) \, ds_1 + \dots \end{aligned} \quad (2.40)$$

and using the relation to the Finger tensor $\mathbf{G}_t := \frac{1}{2}(\mathbf{B}_t - \mathbf{I})$:

$$\begin{aligned} \mathbf{T}_B(t) = & -p\mathbf{I} + \int_0^\infty \mathcal{K}_1(s_1)\mathbf{G}_t(t-s_1) \, ds_1 \\ & + \int_0^\infty \int_0^\infty \mathcal{K}_2(s_1, s_2)\mathbf{G}_t(t-s_1) \cdot \mathbf{G}_t(t-s_2) \, ds_1 + \dots \end{aligned} \quad (2.41)$$

Taking only linear terms into account and choosing the kernel function to be:

$$\mathcal{K}_1(s) := 2\frac{\eta_p}{\lambda_1^2}e^{-\frac{s}{\lambda_1}}, \quad (2.42)$$

we obtain by using $\mathbf{T} = -p\mathbf{I} + \mathbf{S}$ for the extra stress and introducing $t' := t - s$:

$$\mathbf{S}_A(t) = \frac{\eta_p}{\lambda_1}\mathbf{I} - \int_{-\infty}^t \frac{\eta_p}{\lambda_1^2}e^{-\frac{(t-t')}{\lambda_1}}\mathbf{C}_t(t') \, dt', \quad (2.43)$$

and

$$\mathbf{S}_B(t) = -\frac{\eta_p}{\lambda_1}\mathbf{I} + \int_{-\infty}^t \frac{\eta_p}{\lambda_1^2}e^{-\frac{(t-t')}{\lambda_1}}\mathbf{B}_t(t') \, dt' \quad (2.44)$$

which are the integral forms of the Lower Convected Maxwell Model (LCM) and Upper Convected Maxwell Model (UCM). As can be seen on equations (2.43) and (2.44), we can define the positive definite conformation tensors:

$$\mathbf{c}_A(t) := \frac{\eta_p}{\lambda_1}\mathbf{I} - \mathbf{S}_A(t), \quad (2.45)$$

$$\mathbf{c}_B(t) := \frac{\eta_p}{\lambda_1}\mathbf{I} + \mathbf{S}_B(t), \quad (2.46)$$

since the Cauchy-Green tensor \mathbf{C} and the Finger tensor \mathbf{B} are always positive definite. These tensors are used for the momentum transfer equation presented in Sec. 3.3 and can also be discretized instead of the constitutive equation itself, e.g. in the log-conformation method (Kupferman, 2005). Differentiation of (2.44) according to time t by using the relation:

$$\frac{\partial \mathbf{B}_t(t')}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{B}_t(t') + \mathbf{B}_t(t') \cdot \mathbf{L}(t) + \mathbf{L}^T(t) \cdot \mathbf{B}_t(t'). \quad (2.47)$$

leads to:

$$\frac{D\mathbf{S}_B(t)}{Dt} = -\frac{1}{\lambda_1}\mathbf{S}_B(t) + \mathbf{L}^T(t) \cdot \mathbf{S}_B(t) + \mathbf{S}_B(t) \cdot \mathbf{L}(t) + 2\frac{\eta}{\lambda_1}\mathbf{D}(t). \quad (2.48)$$

This is the upper convected Maxwell model:

$$\mathbf{S}_B(t) + \lambda_1 \left(\frac{D\mathbf{S}_B(t)}{Dt} - \mathbf{L}(t) \cdot \mathbf{S}_B(t) - \mathbf{S}_B(t) \cdot \mathbf{L}(t)^T \right) = 2\eta\mathbf{D}(t) \quad (2.49)$$

with the upper convected objective time derivative:

$$\overset{\nabla}{\mathbf{S}} := \frac{\partial \mathbf{S}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{S} - \nabla \mathbf{u} \cdot \mathbf{S} - \mathbf{S} \cdot (\nabla \mathbf{u})^T. \quad (2.50)$$

A proof of Eq.(2.47) and the equivalence of the integral form (Eq. (2.44)) and the differential form (Eq. (2.49)) of the upper convected Maxwell model can be found in Joseph (1990). If we add a Newtonian term to the kernel function (2.42) and choose it to be

$$\mathcal{K}_1(s) := 2 \left(\eta_s \delta(s) + \frac{\eta_p}{\lambda_1} e^{-\frac{s}{\lambda_1}} \right) \quad (2.51)$$

using the Dirac function $\delta(s)$, we obtain the Oldroyd B model:

$$\mathbf{S}(t) + \lambda_1 \overset{\nabla}{\mathbf{S}} = 2\eta_0\mathbf{D} + 2\eta_0\lambda_2 \overset{\nabla}{\mathbf{D}} \quad (2.52)$$

with λ_1 the relaxation time, $\eta_0 := \eta_s + \eta_p$ the total viscosity and $\lambda_2 := \frac{\eta_s\lambda_1}{\eta_s + \eta_p}$ the retardation time (Giesekus, 1994).

The Oldroyd B model is commonly used in numerical studies of viscoelastic flow. It can qualitatively reproduce many features of a so called Boger fluid used in experiments. It predicts correctly in a simple steady shear flow a constant viscosity, the first normal stress difference N_1 which is quadratic in the shear rate, whereas the second normal stress difference N_2 is zero. In the case of unsteady flow, the stresses increase monotonically in time to their steady state values without overshooting, which can be observed in some polymer solutions. In extensional flow the elongational viscosity becomes infinite at a finite elongation rate of $\frac{1}{2\lambda_1}$, which can cause problems, especially in the stability of numerical simulations (Phan-Thien and Mai-Duy, 2017).

If we follow the same strategy described above for the LCM equation (2.43), we would obtain the lower convected Maxwell fluid and the Oldroyd A fluid, respectively, with the lower convected objective derivative:

$$\overset{\Delta}{\mathbf{S}} := \frac{\partial \mathbf{S}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{S} + (\nabla \mathbf{u})^T \cdot \mathbf{S} + \mathbf{S} \cdot \nabla \mathbf{u}. \quad (2.53)$$

Giesekus extended the Maxwell model B to display anisotropic movement of structures such as polymer chains within the flow by introducing a non-linear term for \mathbf{S} :

$$\left(\mathbf{I} + \alpha_G \frac{\lambda_1}{\eta} \mathbf{S} \right) \cdot \mathbf{S} + \lambda_1 \overset{\nabla}{\mathbf{S}} = 2\eta\mathbf{D}. \quad (2.54)$$

2.5. System of Equations for Viscoelastic Single-Phase Flow

In this section, we present the full system which has been discretized using the DG method in the BoSSS framework. We choose the Oldroyd B model for implementation since it is the simplest non-linear rate-typed model displaying most of the physical phenomena such as relaxation and retardation. Furthermore, it is widely used in the literature, e.g., for the benchmark test cases in numerical simulations such as the confined cylinder problem. It can be easily reduced to the Maxwell model B and the Newtonian formulation and it can be easily extended to the Giesekus model which is common in many application-related simulations, like injection moulding (e.g. Gava and Lucchetta, 2012) and other related application-based models (Li et al., 2011).

We consider two-dimensional incompressible fluid flow consisting of the continuity equation

$$\nabla \cdot \mathbf{u} = 0, \quad (2.55)$$

and the momentum equations without body forces

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \nabla \cdot \mathbf{S}. \quad (2.56)$$

The constitutive equations for the Oldroyd B fluid read

$$\mathbf{S} + \lambda_1 \overset{\nabla}{\mathbf{S}} = 2\eta_0 \left(\mathbf{D} + \lambda_2 \overset{\nabla}{\mathbf{D}} \right) \quad (2.57)$$

with the strain rate tensor $\mathbf{D} = \frac{1}{2} \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T \right)$. The constitutive model reduces to the upper convected Maxwell model (Maxwell fluid B) if $\lambda_2 = 0$ and to a Newtonian fluid if both $\lambda_1 = 0$ and $\lambda_2 = 0$.

2.5.1. Splitting into Polymeric and Solvent Part

We want to simplify the system for the numerical simulation by avoiding the upper convected derivative of the strain rate tensor. Therefore, we consider in the following a model of fluids with a polymeric fluid component in a Newtonian solution such that

$$\mathbf{S} := \mathbf{S}_s + \mathbf{S}_p = 2\eta_s \mathbf{D} + \boldsymbol{\tau} \quad (2.58)$$

with

$$\eta_0 := \eta_s + \eta_p. \quad (2.59)$$

Inserted into the momentum equations we obtain

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \eta_s \Delta \mathbf{u} + \nabla \cdot \boldsymbol{\tau}. \quad (2.60)$$

Inserted into the constitutive equations we obtain

$$2\eta_s \mathbf{D} + \boldsymbol{\tau} + 2\eta_s \lambda_1 \overset{\nabla}{\mathbf{D}} + \lambda_1 \overset{\nabla}{\boldsymbol{\tau}} = 2\eta_0 \left(\mathbf{D} + \lambda_2 \overset{\nabla}{\mathbf{D}} \right). \quad (2.61)$$

Using Eq. (2.59) and the identity $\lambda_2 = \frac{\eta_s}{\eta_0} \lambda_1$ we obtain by subtracting terms appearing on both sides

$$\boldsymbol{\tau} + \lambda_1 \overset{\nabla}{\boldsymbol{\tau}} = 2\eta_p \mathbf{D}. \quad (2.62)$$

2.5.2. Non-dimensionalisation

In numerical simulation it is useful to non-dimensionalise the system of equation to easily compare results with different length scales and especially to be able to compare the influence of viscoelasticity by using the Weissenberg number. In the case of viscoelastic flow and physics of droplets we have mostly creeping flow where viscous effects are dominant. In this case, the pressure is non-dimensionalised with $p := \frac{\eta_0 u_c}{l_c} p^\circ$, such that the Reynolds number appears also in front of the pressure and thus, can be brought in front of the material derivative.

We use the following characteristic quantities depending on the setup of a simulation, all denoted with a subscript c: l_c as characteristic length scale, and u_c as characteristic velocity. The non-dimensionalisation for the independent variables reads as follows:

$$\mathbf{x} := l_c \mathbf{x}^\circ, \quad t := \frac{l_c}{u_c} t^\circ, \quad \nabla := \frac{1}{l_c} \nabla^\circ, \quad \Delta := \frac{1}{l_c^2} \Delta^\circ, \quad (2.63)$$

and for the dependent variables:

$$\mathbf{u} := u_c \mathbf{u}^\circ, \quad p := \frac{\eta_0 u_c}{l_c} p^\circ, \quad \boldsymbol{\tau} := \frac{\eta_0 u_c}{l_c} \boldsymbol{\tau}^\circ \quad (2.64)$$

with the dimensionless quantities:

$$\text{Re} := \frac{\rho u_c l_c}{\eta_0}, \quad \text{Wi} := \lambda_1 \frac{u_c}{l_c}, \quad \beta = \frac{\eta_s}{\eta_0}, \quad (1 - \beta) := \frac{\eta_p}{\eta_0}, \quad (2.65)$$

which leads to the following non-dimensionalised system of equations:

$$\nabla^\circ \cdot \mathbf{u}^\circ = 0, \quad (2.66)$$

$$\text{Re} \frac{D\mathbf{u}^\circ}{Dt^\circ} = -\nabla^\circ p^\circ + \beta \Delta^\circ \mathbf{u}^\circ + \nabla^\circ \cdot \boldsymbol{\tau}^\circ, \quad (2.67)$$

$$\boldsymbol{\tau}^\circ + \text{Wi} \overset{\nabla^\circ}{\boldsymbol{\tau}^\circ} = 2(1 - \beta) \mathbf{D}^\circ. \quad (2.68)$$

In index notation the system reads

$$\frac{\partial u_i^\circ}{\partial x_i^\circ} = 0, \quad (2.69)$$

$$\text{Re} \left(\frac{\partial u_i^\circ}{\partial t^\circ} + u_j^\circ \frac{\partial u_i^\circ}{\partial x_j^\circ} \right) = -\frac{\partial p^\circ}{\partial x_i^\circ} + \beta \frac{\partial^2 u_i^\circ}{\partial x_j^2} + \frac{\partial \tau_{ij}^\circ}{\partial x_j^\circ}, \quad (2.70)$$

$$\tau_{ij}^\circ + \text{Wi} \left(\frac{\partial \tau_{ij}^\circ}{\partial t^\circ} + u_k^\circ \frac{\partial \tau_{ij}^\circ}{\partial x_k^\circ} - \frac{\partial u_i^\circ}{\partial x_k^\circ} \tau_{kj}^\circ - \tau_{ik}^\circ \frac{\partial u_j^\circ}{\partial x_k^\circ} \right) = (1 - \beta) \left(\frac{\partial u_i^\circ}{\partial x_j^\circ} + \frac{\partial u_j^\circ}{\partial x_i^\circ} \right). \quad (2.71)$$

In the following, we omit the superscript of the non-dimensionalised quantities such that the quantities used in the rest of this work are free of any superscript for better readability.

Remark: Non-dimensionalisation of the Pressure

The non-dimensionalisation of the pressure presented above is chosen for creeping flow where viscous effects are dominant. In case of high velocity flows where dynamic effects are dominant, the pressure can be non-dimensionalised with $p := \rho u_c^2 \bar{p}$, such that the Reynolds number appears in front of the viscous terms and the momentum equation reads:

$$\frac{D\mathbf{u}}{Dt} = -\nabla p + \frac{\beta}{\text{Re}} \Delta \mathbf{u} + \frac{1}{\text{Re}} \nabla \cdot \boldsymbol{\tau} \quad (2.72)$$

or in index notation

$$\left(\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} \right) = -\frac{\partial p}{\partial x_i} + \frac{\beta}{\text{Re}} \frac{\partial^2 u_i}{\partial x_j^2} + \frac{1}{\text{Re}} \frac{\partial \tau_{ij}}{\partial x_j}. \quad (2.73)$$

In this work, we use the non-dimensionalisation for creeping flow if not stated differently.

Remark: Non-dimensionalisation of the Relaxation Time

The relaxation time λ_1 has been non-dimensionalised using the Weissenberg number Wi shown in Eq. (2.65). The Weissenberg number relates the elastic to viscous forces such that a large Weissenberg number describes significant non-Newtonian behaviour if the elastic forces outweigh the viscous forces.

A second dimensionless number to non-dimensionalise the relaxation time is the Deborah-number De defined as follows:

$$De := \frac{\lambda_1}{t_c} = \lambda_1 \omega_c. \quad (2.74)$$

It describes the ratio between the fluid relaxation time and a characteristic time (t_c) or frequency (ω_c) where often an observation time scale is chosen. This scaling has more the characteristic of describing the transient behaviour of the fading memory relative to the fluid time scale. Hence, the difference between fluid and solid is seen as a difference between observation time scales. This means that for small Deborah numbers (large observation time scale) the continuous body behaves more fluid-like and for large Deborah numbers (small observation time scales) it shows more solid-like behaviour. In the limits this means for $De = 0$ we have a Newtonian fluid and for $De = \infty$ we have an elastic solid.

Both dimensionless numbers are found in literature strongly depending on the test case or experiment. The relation between those numbers is displayed in the Pipkin diagram (fig. 2.1). In some cases, both numbers are equal and can be used equally, this is the case e.g., if the characteristic observation time is defined by the characteristic length divided by the characteristic velocity.

In this work, we use the Weissenberg number for non-dimensionalisation if not stated differently.

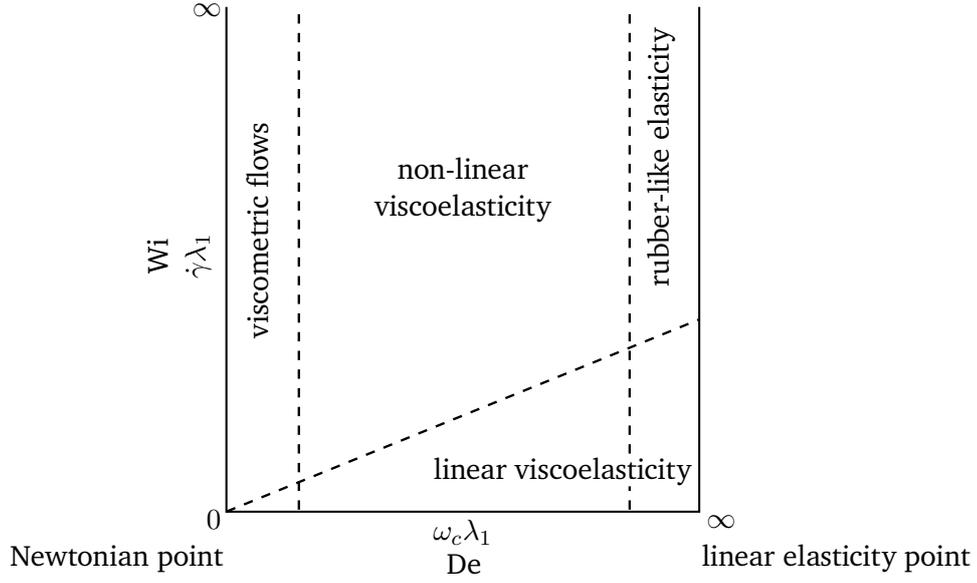


Figure 2.1.: Pipkin diagram showing the relation between Weissenberg and Deborah number (Phan-Thien and Mai-Duy, 2017). The regions beyond the dashed lines show linear behaviour: For low Deborah numbers this is viscometric flow, for high Deborah numbers this is rubber-like elasticity. For low Weissenberg numbers the viscoelastic behaviour is linear for all Deborah numbers resulting in linear elasticity if $De \rightarrow \infty$. If both numbers are zero, the flow is Newtonian.

2.6. The Viscoelastic Multi-Phase Setting

For the two-phase setting of our problem, we have to define a partitioning with the two disjoint phases which we denote as phase \mathfrak{A} and phase \mathfrak{B} , and an interface of the two phases \mathfrak{I} within the domain of interest $\Omega \subset \mathbb{R}^2$:

$$\Omega := \mathfrak{A}(t) \dot{\cup} \mathfrak{I}(t) \dot{\cup} \mathfrak{B}(t). \quad (2.75)$$

We consider the non-dimensionalised system of equations (2.66) - (2.68) for both phases in the bulk $\Omega \setminus \mathfrak{I}$ with piecewise constant material parameters in \mathfrak{A} and \mathfrak{B} :

$$\text{Re}(\mathbf{x}) := \begin{cases} \text{Re}_{\mathfrak{A}}, & \mathbf{x} \in \mathfrak{A} \\ \text{Re}_{\mathfrak{B}}, & \mathbf{x} \in \mathfrak{B} \end{cases}, \quad \text{Wi}(\mathbf{x}) := \begin{cases} \text{Wi}_{\mathfrak{A}}, & \mathbf{x} \in \mathfrak{A} \\ \text{Wi}_{\mathfrak{B}}, & \mathbf{x} \in \mathfrak{B} \end{cases}, \quad \beta(\mathbf{x}) := \begin{cases} \beta_{\mathfrak{A}}, & \mathbf{x} \in \mathfrak{A} \\ \beta_{\mathfrak{B}}, & \mathbf{x} \in \mathfrak{B} \end{cases}. \quad (2.76)$$

We consider the fluid interface $\mathfrak{I}(t)$ to be a one-dimensional manifold which has at least a continuous and globally bounded curvature (Kummer, 2017). For the continuity (2.66) and momentum equation (2.67) we define jump conditions valid on a material interface with no-slip boundary conditions:

$$[[\mathbf{u}]] = 0, \quad (2.77)$$

$$[[p \mathbf{n}_{\mathfrak{I}} - \beta (\nabla \mathbf{u} + (\nabla \mathbf{u})^T) \cdot \mathbf{n}_{\mathfrak{I}} - \boldsymbol{\tau} \cdot \mathbf{n}_{\mathfrak{I}}]] = \frac{1}{\text{We}} \kappa \mathbf{n}_{\mathfrak{I}} \quad (2.78)$$

with We the Weber number, $\kappa = \nabla \cdot \mathbf{n}_{\mathcal{I}}$ the mean curvature of the interface \mathcal{I} and $\mathbf{n}_{\mathcal{I}}$ the normal field on the interface which is showing from the phase \mathfrak{A} to \mathfrak{B} . The jump operator $\llbracket - \rrbracket$ is defined as follows:

$$\llbracket \mathbf{u} \rrbracket(\mathbf{x}) := \lim_{\varepsilon \searrow 0} (\mathbf{u}(\mathbf{x} + \varepsilon \mathbf{n}_{\mathcal{I}}) - \mathbf{u}(\mathbf{x} - \varepsilon \mathbf{n}_{\mathcal{I}})). \quad (2.79)$$

for $\mathbf{x} \in \mathcal{I}$ (Kummer, 2017).

The Weber number is defined as follows:

$$We := \frac{\hat{\rho} u_c^2 l_c}{\sigma} \quad (2.80)$$

with σ the surface tension coefficient and the constant density across the interface $\hat{\rho} = \frac{1}{2}(\rho_{\mathfrak{A}} + \rho_{\mathfrak{B}})$. The Weber number denotes the relative value of the inertial to the surface tension forces (Gross and Reusken, 2011).

For the constitutive equation (2.68), there exists no jump condition since this only describes local material properties, which may not affect the dependent variables at the interface and which are only displayed by the jumps in the piecewise constant material parameters (2.76). However, the description of the momentum balance of the interface with a constant surface tension σ does not properly describe the physical behaviour for an interface with viscoelastic behaviour involved. Therefore, it is useful to introduce dynamic parts into the surface stress force in a more general ansatz. We consider the surface tension to be a contact force on each surface segment $l_s \subset \mathcal{I}$:

$$F_c := \int_{\partial l_s} \mathbf{T}_{\mathcal{I}} \mathbf{n} \, dS = \oint_{l_s} \nabla_{\mathcal{I}} \cdot \mathbf{T}_{\mathcal{I}} \, dl_s \quad (2.81)$$

with $\nabla_{\mathcal{I}} = \mathbf{P}_{\mathcal{I}} \nabla$ and $\mathbf{P}_{\mathcal{I}} = \mathbf{I} - \mathbf{n}_{\mathcal{I}} \otimes \mathbf{n}_{\mathcal{I}}$ the orthogonal projection on \mathcal{I} . The projection is used since only contact forces, which are tangential to the surface, should be taken into account (Gross and Reusken, 2011). We can decompose the surface tension force into an isotropic and a deviatoric part analogously to the Cauchy stress tensor defined in Eq. (2.23):

$$\mathbf{T}_{\mathcal{I}} = \sigma \mathbf{P}_{\mathcal{I}} + \mathbf{T}_{\mathcal{I}}^D. \quad (2.82)$$

We now want to give this force in a first step a simple linear physical behaviour. This is why we introduce the Boussinesq-Scriven model for a variable surface tension force which enables the interface to show inertia. This model is based on the idea to consider the interface as a Newtonian fluid (Gross and Reusken, 2011, e.g.). Since for the interface we cannot assume $\text{tr}(\mathbf{D}) = \nabla \cdot \mathbf{u} = 0$ like in the bulk phases, we have to involve the volumetric change at the interface. Hence, we start analogously to Eq. (2.27) with:

$$\mathbf{T}_{\mathcal{I}} = \sigma \mathbf{P}_{\mathcal{I}} + \mathcal{L}(\mathbf{D}_{\mathcal{I}}(\mathbf{u})), \quad \mathbf{D}_{\mathcal{I}}(\mathbf{u}) = \mathbf{P}_{\mathcal{I}} \cdot (\nabla_{\mathcal{I}} \mathbf{u} + (\nabla_{\mathcal{I}} \mathbf{u})^T) \cdot \mathbf{P}_{\mathcal{I}}, \quad (2.83)$$

where \mathcal{L} is a linear operator. If we now consider the principles presented in section 2.4.2, we obtain the interface stress tensor:

$$\mathbf{T}_{\mathcal{I}} = \sigma \mathbf{P}_{\mathcal{I}} + \tilde{\Lambda}_{\mathcal{I}} (\nabla_{\mathcal{I}} \cdot \mathbf{u}) \mathbf{P}_{\mathcal{I}} + \eta_{\mathcal{I}} \mathbf{D}_{\mathcal{I}}(\mathbf{u}). \quad (2.84)$$

As mentioned previously, $\nabla_{\mathcal{I}} \cdot \mathbf{u} \neq 0$, although $\nabla \cdot \mathbf{u} = 0$. It can be useful to further restrict $\tilde{\Lambda}_{\mathcal{I}}$ e.g. to be $\tilde{\Lambda}_{\mathcal{I}} > -\eta_{\mathcal{I}}$, such that it is redefined with $\tilde{\Lambda}_{\mathcal{I}} = \Lambda_{\mathcal{I}} - \eta_{\mathcal{I}}$ and $\Lambda_{\mathcal{I}} = \tilde{\Lambda}_{\mathcal{I}} + \eta_{\mathcal{I}} > 0$

where $\Lambda_{\mathcal{I}}$ is the interface dilatant viscosity and $\eta_{\mathcal{I}}$ is the interface shear viscosity (Gross and Reusken, 2011).

Thus, the non-dimensionalised Boussinesq-Scriven model for the jump condition of the momentum equation at the interface (2.78) reads:

$$\llbracket p \mathbf{n}_{\mathcal{I}} - \beta \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T \right) \cdot \mathbf{n}_{\mathcal{I}} - \boldsymbol{\tau} \cdot \mathbf{n}_{\mathcal{I}} \rrbracket = \nabla_{\mathcal{I}} \cdot \left(\frac{1}{\text{We}} \mathbf{P}_{\mathcal{I}} + \frac{\tilde{\Lambda}_{\mathcal{I}}}{\eta_0} (\nabla_{\mathcal{I}} \cdot \mathbf{u}) \mathbf{P}_{\mathcal{I}} + \frac{\eta_{\mathcal{I}}}{\eta_0} \mathbf{D}_{\mathcal{I}}(\mathbf{u}) \right). \quad (2.85)$$

This model could be extended in future with a non-linear relation and time-dependent viscoelastic-like relation to $\mathbf{D}_{\mathcal{I}}(\mathbf{u})$. Note that unsmooth geometries of the interface can lead to an ill-posed Boussinesq-Scriven model (Bothe and Prüss, 2010):

3. The High Weissenberg Number Problem

A breakdown in convergence for different discretization schemes for low Weissenberg numbers of order $\mathcal{O}(1)$ was already observed in the early years of computational methods solving viscoelastic flow problems. This phenomenon which is widely discussed in the scientific community is called the High Weissenberg Number Problem (HWNP). With more and more understanding about the problems arising using viscoelastic constitutive equations for simulation, many strategies and stabilization schemes were developed over time, such that nowadays simulations for $Wi = \mathcal{O}(10)$ are possible (e.g. Hulsen et al., 2005; Comminal et al., 2015; Niethammer et al., 2018; Mu et al., 2019). Since there are few analytical solutions, a couple of benchmark problems were developed for comparing results and measuring the achievable Weissenberg numbers. These problems are usually with sharp corners, narrowings and stagnation points such that there are steep gradients with a strong viscoelastic response in the flow. The most popular benchmark problems are a confined cylinder immersed in a narrow channel with a blocking ratio with the channel height to the cylinder radius of 2 : 1 (e.g. Claus and Phillips, 2013), a 4 : 1 contraction with a channel sharply narrowing from height 4 to 1 and a singularity at the corner (e.g. Kim et al., 2005), a stick-slip problem (e.g. Baaijens, 1994b), and a lid-driven cavity (e.g. Comminal et al., 2015).

In this chapter we want to give an overview over the state of art in numerical simulation of viscoelastic flow. First, we explain the mathematical characteristics of the constitutive equations. Afterwards, we shortly name different numerical methods used in context of viscoelastic flow. Then, we present in historically order the most common methods to circumvent stability issues arising from the numerical behaviour of the constitutive laws. Especially the newest log-conformation method which treats different instabilities than the other methods mentioned shows high potential for a wide range of different benchmark problems to achieve high Weissenberg numbers (e.g. Fattal and Kupferman, 2005; Hulsen et al., 2005; Comminal et al., 2015; Mu et al., 2019). In the end we explain the most common stabilization methods using upwinding for the convection dominated constitutive equations.

Since we use a different ansatz with a solely high-order DG method and are capable to discretize hyperbolic constitutive equations by using the LDG formulation, we are not using most of the techniques described in this chapter. Although the log-conformation method is promising to defeat the HWNP, we also do not make use of it. This is because the focus of this solver is not to simulate the different benchmark problems with high Weissenberg numbers, but to obtain results for droplets in viscoelastic flow where usually no sharp changes in flow and exponential stress profiles are expected. However, results for the confined cylinder benchmark problem to validate the presented single-phase solver can be found in Chap. 7.

3.1. Mathematical Aspects of Viscoelastic Models

We consider a two-dimensional differential equation of second order:

$$\begin{aligned} f_1(x, y) \frac{\partial^2 \psi(x, y)}{\partial x^2} + f_2(x, y) \frac{\partial^2 \psi(x, y)}{\partial x \partial y} + f_3(x, y) \frac{\partial^2 \psi(x, y)}{\partial y^2} \\ + f_4(x, y) \frac{\partial \psi(x, y)}{\partial x} + f_5(x, y) \frac{\partial \psi(x, y)}{\partial y} + f_6(x, y) \psi(x, y) = 0. \end{aligned} \quad (3.1)$$

This partial differential equation can be of different type with impacts on its physical behaviour. We define:

$$D(x, y) := f_1(x, y)f_3(x, y) - \left(\frac{f_2(x, y)}{2} \right)^2. \quad (3.2)$$

If $D(x, y) > 0$ equation (3.1) turns into an elliptic problem in (x, y) . For $D(x, y) = 0$ we have a parabolic type and for $D(x, y) < 0$ the equation is hyperbolic. This condition and thus, the labelling is deduced from the conic section equation where the condition leads to corresponding section of that specific form. The most prominent and simplest examples are the Laplace ($\Delta\psi = 0$) or Poisson equation ($\Delta\psi = f$) for the elliptic type, the heat equation $\frac{\partial \psi}{\partial t} = a\Delta\psi$ for the parabolic case, and the hyperbolic wave equation $\frac{\partial^2 \psi}{\partial t^2} = c_s^2 \frac{\partial^2 \psi}{\partial x^2}$. Special solution to such problems can be found by introducing appropriate Initial Condition (IC) and BC to gain a well-posed problem, e.g. BC of Dirichlet and Neumann type for the elliptic problem, IC and BC for the parabolic problem and IC for the hyperbolic case.

Using the BC and IC we can reformulate a parabolic or hyperbolic problem into a second order Cauchy problem with:

$$\psi' = f_1(t, \psi), \quad \psi'' = f_2(t, \psi, \psi'), \quad (3.3)$$

where a solution can be found for example, if we know conditions for ψ and ψ' for $t = 0$. Hence, the Cauchy problem is an evolutionary equation for an initial value or mixed initial boundary value problem. This evolution character simplifies the numerical solution because a stepwise solution for sufficiently small Δt starting from the IC can be found. Note, that there can exist a steady state solution for evolutionary problems for $t \rightarrow \infty$, e.g. the Laplace equation can be a steady state solution of the heat equation satisfying the same BC, in this case the evolution equation stabilizes on a solution.

If a Cauchy problem such as Eq. (3.3) is ill-posed, which means that a solution is inexistent, non-unique or the solution's behaviour changes with the IC, or if it is even ill-conditioned it can lose its evolutionary character and instabilities arising in a numerical solution, e.g. short waves of an hyperbolic equation increase sharply in amplitude.

If we now consider a system of equations describing the flow of a viscoelastic fluid we have a system of mixed hyperbolic-elliptic-parabolic type, so changes from elliptic to hyperbolic can occur in some regions of the flow and trigger a loss of evolution. In a series of works (e.g. Joseph and Saut, 1986; Joseph et al., 1987; Joseph, 1990) Joseph and Saut examined this behaviour for viscoelastic fluids with a general example constitutive equation of the form:

$$\lambda_1 \overset{\circ}{\boldsymbol{\tau}} = 2\eta_p \mathbf{D} + \mathbf{J} \quad (3.4)$$

with the corotational Jaumann derivative $\overset{\circ}{\boldsymbol{\tau}} := \frac{\partial \boldsymbol{\tau}}{\partial t} + \mathbf{u} \nabla \cdot \boldsymbol{\tau} + \boldsymbol{\tau} \cdot \mathbf{W} - \mathbf{W} \cdot \boldsymbol{\tau} - a(\mathbf{D} \cdot \boldsymbol{\tau} + \boldsymbol{\tau} \cdot \mathbf{D})$. The term \mathbf{J} consists of low-order terms depending on $\boldsymbol{\tau}$ and \mathbf{u} , but not on their derivatives.

This model describes a viscoelastic fluid with instantaneous elasticity without retardation time $\lambda_2 = 0$ and if the terms are chosen $\mathbf{J} := -\boldsymbol{\tau}$ and $a = [1, 0, -1]$ we have the UCM, the corotational Maxwell fluid and the LCM.

The unsteady quasilinear system of equation describing flow using this model (eq. (3.4)) is evolutionary if the corresponding Cauchy problem is well-posed. However, a change of type and a loss of evolution can occur for specific flow problems. To analyse the behaviour in detail, Joseph and Saut (1986) consider two aspects in stability analysis. First, they use a classical perturbation method by introducing a decomposition of the dependent variables into a basis solution of the system and a small disturbance of it. They concentrate on short waves of propagation assuming that the basis solution and derivatives are constant. They obtain a velocity of propagation c_s of the wave fronts with $c_s^2 = f$ and f real valued. If $f > 0$ the waves propagate with c_s in the field, however, if $f < 0$ then $c_s = \pm i\sqrt{|f|}$ and there exist short waves of rapidly growing amplitude, the Hadamard instabilities described above. The stability to short wave disturbances is reached, if the eigenvalues of the stress disturbance $\hat{\tau}_1 \geq \hat{\tau}_2 \geq \hat{\tau}_3$ are satisfying for a 3D problem:

$$\frac{\eta_p}{\lambda_1} + \frac{1}{2}a(\hat{\tau}_1 + \hat{\tau}_3) - \frac{1}{2}(\hat{\tau}_1 - \hat{\tau}_3) > 0. \quad (3.5)$$

Second, Joseph and Saut (1986) consider the corresponding vorticity equation for steady inertia-less flow which can be either elliptic or hyperbolic and can change regionally, e.g. in case of transonic flow. For two-dimensional flow, there are six characteristics to the problem: two are always imaginary, two are real and correspond to the streamlines of the flow and two can be either real or complex. The latter are associated with the vorticity equation. Here, they found out that a loss of evolution must occur for the unsteady flow case, if the vorticity equation for the corresponding steady inertia-less flow changes from elliptic to hyperbolic, say, if in some regions of the flow the last two characteristics change from real to imaginary.

There are some models such as the Newtonian or the UCM and the LCM which never change type and for which the vorticity equation always stays elliptic. Those findings lead to discussions if models which lose evolution should not be considered any longer. However, the loss of evolution can give a hint to physical instabilities such as melt fracture.

The loss of evolution can also occur for stable systems due to numerical errors and can have, either induced by the model or by numerical errors, a great impact on the success and failure of a numerical simulation, especially for increasing Weissenberg numbers. This leads to several approaches to avoid Hadamard and other instabilities in the flow which will be presented in Sec. 3.3.

3.2. Overview over Discretization Methods

When discretizing a system of equations, the aim is to find a convergent discrete approximation of the solution on a mesh and the discrete problem needs to be solved accurately and in terms of computational time efficiently. As we have already seen before, the discretization of the system of equations for viscoelastic flow is particularly difficult due to the mixed elliptic-hyperbolic type of the equations with convection-dominated behaviour of the hyperbolic constitutive equations.

Conventional discretization methods such as the FD method and FEM are originally developed and profoundly theoretically analyzed for solving elliptic systems of equations, but though unstable for convection-diffusion problems with too coarse meshes. In these methods some kind of artificial diffusivity is often needed to balance the dominance of the convective terms or a reformulation of the governing equations towards an elliptic system (Owens and Phillips, 2002). Such methods for stabilizing the convective terms or the mixed hyperbolic-elliptic-parabolic nature are presented in the next sections.

Discretization methods used in literature for the discretization of the viscoelastic system of equations are the FD and the FV method, FEM and spectral methods. In the context of viscoelastic flow, the methods are often high-order if possible. The spectral method is a global method where the information over the whole domain or subdomain is approximated and therefore, the discretization parameter is the degree of approximation spaces k . The other methods have the triangulation of the domain with suitable elements and the approximation of the solution on a local level in common. Thus, their discretization parameter is the mesh size h . The structure of the algebraic system to solve after linearization is a banded matrix for the local methods and a non-sparse one for the global method. While the spectral method is superior in case of smooth problems concerning accuracy, the accuracy of a local approximation is increased by mesh refinement (increased size of the system) or by using higher-order approximation spaces (increased band width; Owens and Phillips, 2002).

In the FD the derivatives of the dependent variables are approximated by a Taylor expansion such as a second order central difference. The system of equations is mostly reformulated in terms of the stream function, vorticity and extra-stresses. Two sorts of errors occur by neglecting terms of the expansion of order $\mathcal{O}(h^2)$ and greater and by the residual emerging in the numerical approximation. An upwinding scheme stabilizes the convective terms in the extra-stresses, but introduces non-physical diffusion which can dominate the true diffusion term. The central differencing has an error of order $\mathcal{O}(h^2)$ and the upwinding scheme an error of order $\mathcal{O}(h)$. One problem is the lack of an explicit BC for the vorticity equations (Owens and Phillips, 2002). Newer Examples for the use of this method are the works of Fattal and Kupferman (2005) and Comminal et al. (2015).

The main advantage of the FV method is the approach to preserve conservativity of the discrete system by applying the discretization to the conservative form $\nabla \cdot (\cdot) = 0$. The conservation equations are integrated over a control volume and finite difference approximations are used to approach the line integrals over each side of the control volume. Analogously to the FD method, an upwinding scheme can be used for convective terms. This causes the distribution of the transport properties of the flow to smear out, especially, if the flow is not in alignment with the grid lines of the chosen mesh. This can be counteracted by using high-order upwind approximations for the convective fluxes. In a special version of the FV method, the semi-Lagrangian method, the convection and diffusion terms are decoupled from each other (Owens and Phillips, 2002). Examples for the FV method are the code of the group around Phan-Thien (Xue et al., 1995; Dou and Phan-Thien, 2007) and the work of Oliveira and Miranda, 2005 and particularly for the semi-Lagrangian FV method Phillips and Williams (1999).

A very widely used method in the context of viscoelastic flow is the FEM. In this method a variational weak formulation of the equations is found. Let $\mathcal{V} \times \mathcal{Q} \times \mathcal{S}$ be appropriate, but not specified function spaces for velocity, pressure and extra-stress, then the weak form reads:

find $(\mathbf{u}, p, \boldsymbol{\tau}) \in \mathcal{V} \times \mathcal{Q} \times \mathcal{S}$

$$\begin{aligned} \int_{\Omega} \nabla \cdot \mathbf{u} q \, dV &= 0, \quad \forall q \in \mathcal{Q}, \\ \int_{\Omega} (\beta(\nabla \mathbf{u})^T : \nabla \mathbf{v} + \boldsymbol{\tau} : \nabla \mathbf{v} - p \nabla \cdot \mathbf{v} \, dV &= 0, \quad \forall \mathbf{v} \in \mathcal{V}, \\ \int_{\Omega} (\boldsymbol{\tau} + \text{Wi} \overset{\nabla}{\boldsymbol{\tau}} - (1 - \beta) \mathbf{D}) : \boldsymbol{\sigma} \, dV &= 0, \quad \forall \boldsymbol{\sigma} \in \mathcal{S}. \end{aligned} \quad (3.6)$$

The discretization spaces for the test and trial functions contain certain polynomials of certain order in each element with a given degree of continuity between cells. Mostly, the Galerkin method is used, where the same basis function are applied to the test and the solution space. Hierarchical basis functions as hat functions of e.g. piecewise linear approximation spaces are chosen. The main advantage of this method is its wide adaptability to irregular geometries or complicated BC. The algebraic system contains a mass and a stiffness matrix, where the contributions can be calculated at elemental level using a local coordinate system and the global system requires information about the mapping from the local to global coordinates (Owens and Phillips, 2002).

In case of viscoelastic flow there are two main issues. First, by the choice of appropriate approximation spaces for the velocity, pressure and extra stresses certain compatibility conditions for the well-posedness of the system have to be met. One condition is the Ladyzenskaja-Babuska-Brezzi (LBB) condition which has been studied theoretically e.g. for the velocity-pressure formulation of the Stokes problem (Babuška, 1973): There exist a constant $a > 0$ such that:

$$\inf_{q_h \in \mathcal{Q}} \sup_{\mathbf{u}_h \in \mathcal{V}} \frac{(\nabla_h \cdot \mathbf{u}_h, q_h)}{\|\mathbf{u}_h\|_{H^1(\Omega_h)} \|q_h\|_{L^2(\Omega_h)}} \geq a \quad (3.7)$$

The existence and appearance of such conditions for the extra-stresses in the viscoelastic case is not well examined. The second is the problem of handling the convection dominated nature of the constitutive equations. While the standard Galerkin approach is perfect for elliptic problems and possesses the best approximation properties these are lost in the hyperbolic case. Spurious oscillations can occur for non-smooth solutions, especially, if the mesh is too coarse for resolving thin boundary layers, and are due to a lack of stability. Also here, upwinding techniques with a loss of accuracy are used to stabilize the discretization (Owens and Phillips, 2002).

Some examples for FEM from different decades are Talwar and Khomami (1992), Kim et al. (2004), Kim et al. (2005), Hulsén et al. (2005), and Kynch and Phillips (2017) and the work of Fortin (Fortin and Fortin, 1989; Fortin and Fortin, 1990; Fortin and Zine, 1992). Newest publications are Mu et al. (2019) and Varchanis et al. (2019). There are also hybrid FV and FEM schemes, for example a working group around Webster developed a code on this basis (Wapperom and Webster, 1998; Aboubacar and Webster, 2001; Aboubacar and Webster, 2003).

In the spectral methods the discrete setting contains global information using high-order approximations over the whole domain which should be rectangular or of a standard shape. Therefore, periodic functions are expanded using Fourier transformation while non-periodic functions can be expanded by the eigenfunctions of a singular Sturm-Liouville problem. This can be Chebychev or Legendre polynomials. The rate of decay of the expansion coefficients is solely determined by the smoothness of the function and not by any special conditions satisfied

by the function at the boundary. If Legendre polynomials are used, the expansion is based on either Lagrange interpolates or orthogonal polynomials.

If the computational domain differs from standard shape, there is either the possibility to map the domain or subdomains onto a rectangular shape with an appropriate coordinate transformation or to combine the spectral method with the FEM by discretizing the weak formulation of the system (3.6) and choosing Lagrange interpolates instead of hierarchical basis functions. The latter has a high flexibility because elements can be concentrated in certain parts of the domain where steep gradients of the stresses need to be resolved.

Pure spectral computational methods without the FEM strategy are older, e.g. Pilitsis et al. (1991) and Sureshkumar and Beris (1995). The combination of FEM and spectral methods was strongly emerging (e.g. Fiétier and Deville, 2003), here the LUST algorithm of Owens et al. (2002) should be mentioned (Chauvière and Owens, 2000; Chauvière and Owens, 2001) and the newer work of Claus and Phillips (2013).

3.3. Stabilization Methods for the Coupled System

As we have seen in Sec. 3.1, it is very likely that for different flow problems and a variety of viscoelastic models a change of type from elliptic to hyperbolic can occur. A Newtonian solvent which is added to most differential viscoelastic constitutive equations mathematically regularizes the coupled system of equations formed by the momentum, continuity and constitutive equations. The momentum and continuity equations form an elliptic saddle-point problem for velocity and pressure and the constitutive equation is hyperbolic in stress (Rajagopalan et al., 1990).

Thus, a standard process for the Oldroyd B fluid is to decompose the stresses into a Newtonian solvent and a polymeric part using the ansatz as presented in Sec. 2.5.1. As a result, the momentum equation stays always elliptic as long as we have an inertia-driven flow and the solvent viscosity is large ($\beta \rightarrow 1$). All hyperbolic parts are in the constitutive equation for the polymeric stress and, as long as their contribution is small, they can be discretized separately without influencing the elliptic momentum equation. This works especially for decoupled iterative schemes. This viscous splitting makes also sense in a physical point of view because the numerical simulations, especially for the Oldroyd B fluid, are often conducted for physical test-cases of a polymer solution with polymeric molecules distributed in a Newtonian solvent (Owens and Phillips, 2002).

If we now consider fluids like the UCM and the LCM with an immediate elastic response, which means that there is no retardation ($\lambda_2 = 0$) and thus, the solvent viscosity tends to zero ($\eta_s = 0$ or $\beta = 0$), the elliptic operator in the momentum equations becomes singular and a change of type to the hyperbolic constitutive equations can occur. This can be analyzed considering the momentum transport or momentum transfer for the UCM:

$$\chi := \text{Wi } \boldsymbol{\tau} + \boldsymbol{I} - \text{Re Wi } \boldsymbol{u}\boldsymbol{u}. \quad (3.8)$$

The momentum transfer becomes hyperbolic if the sign of the determinant of χ changes from positive to negative and is thus no longer positive definite. This might occur, if the Reynolds number is increased for a fixed Weissenberg number. As already stated in section 3.1, the system for the UCM and LCM will never change type for inertia-less flow ($\text{Re} = 0$)

since the conformation tensor, here for the UCM ($c_B = Wi \boldsymbol{\tau} + \mathbf{I}$), is always positive definite (Rajagopalan et al., 1990; Joseph et al., 1987; Owens and Phillips, 2002).

There are several methods described in literature to handle this problem in avoiding the system to become hyperbolic. These methods are described in their historical order of appearance in the following Sec. 3.3.1-3.3.3. However, in the scope of this work, we use the DG method which is a very robust method for handling hyperbolic conservation laws and while using the LDG method which stabilizes the coupling to the hyperbolic constitutive equations, we have no need to introduce such stabilizing methods avoiding the system to become hyperbolic.

3.3.1. Explicitly Elliptic Momentum Equation Method

The idea of the Explicitly Elliptic Momentum Equation (EEME) method is to obtain a second order elliptic equation out of the system of first order hyperbolic equations. Therefore, the divergence of the polymeric stress is inserted into the inertia-less momentum equation with a modified pressure. The resulting momentum equation is an always elliptic second order equation for all Deborah numbers in case of the UCM. It was first introduced by Renardy (1985) for a proof of existence for steady creeping viscoelastic flow. Later on, King et al. (1988) firstly used the equation in the numerical context for simulating the stick-slip benchmark problem. The equation is constructed as follows: Initially, the divergence of the polymeric stress of the Oldroyd B model is taken (Rajagopalan et al., 1990):

$$\nabla \cdot \mathbf{S}_p = -De \mathbf{u} \cdot \nabla (\nabla \cdot \mathbf{S}_p) + \nabla \cdot [(\text{De } \mathbf{S}_p - (1 - \beta) \mathbf{I}) \cdot \mathbf{u}]. \quad (3.9)$$

This is substituted into the inertia-less momentum equation $\nabla \cdot \mathbf{S}_p = \nabla p$ by using the total stress having $\mathbf{S}_p = \mathbf{S} + \beta \mathbf{D}$ and by defining the modified pressure $p_m = p + De \mathbf{u} \cdot \nabla p$:

$$\nabla \cdot [(\text{De } \mathbf{S}_p - \mathbf{I}) \cdot \nabla \mathbf{u}] + De (\nabla \mathbf{u}) \cdot (\nabla \cdot \mathbf{S}_p) + \nabla p_m - De \beta [(\nabla \mathbf{u}) \cdot (\Delta \mathbf{u}) + \mathbf{u} \cdot \nabla (\Delta \mathbf{u})] = 0. \quad (3.10)$$

Although this formulation does not have second order derivatives in the stresses, there are third order derivatives of the velocities which can have singular effects on the equation for large solvent viscosities ($\beta \rightarrow 1$; Owens and Phillips, 2002). For the UCM ($\beta = 0$) the eq. (3.10) can be reformulated using the momentum transfer (3.8) for inertia-less flow:

$$\nabla \cdot \text{De} (\boldsymbol{\chi} (\nabla \mathbf{u})) + \text{De} (\nabla \cdot \boldsymbol{\chi}) (\nabla \mathbf{u})^T = \nabla p_m \quad (3.11)$$

where in the inertia-less case the momentum transfer is identical with the conformation tensor $\boldsymbol{\chi} = c_B$ and since the conformation tensor is always positive definite, the EEME for the inertia-less UCM is stable for all Deborah numbers Owens and Phillips (2002). In case of Oldroyd B flow, the EEME must also be stable for small $\beta \ll 1$ and for the UCM model the EEME equation can be extended to flow with inertia using the momentum transfer in eq. (3.8) and inserting into Eq. (3.11) (Rajagopalan et al., 1990). For the numerical discretization of the EEME, special attention has to be paid on the higher order terms since in continuous approximations such as the Galerkin method these are not well defined.

3.3.2. Elastic Viscous Stress Splitting Type Discretization

The idea of EVSS is to add an elliptic Newtonian operator to the hyperbolic constitutive equations for the polymeric stress such that the modified polymeric stress reads (Rajagopalan et al., 1990):

$$\boldsymbol{\Sigma} := \boldsymbol{\tau} - (1 - \beta)\mathbf{D}. \quad (3.12)$$

Inserting this into the system of equations for Oldroyd B flow (2.66-2.68), we obtain the following system for steady flow:

$$\nabla \cdot \mathbf{u} = 0, \quad (3.13)$$

$$\text{Re}(\mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla p + \Delta \mathbf{u} - \nabla \cdot \boldsymbol{\Sigma}, \quad (3.14)$$

$$\boldsymbol{\Sigma} + \text{Wi} \overset{\nabla}{\boldsymbol{\Sigma}} = (1 - \beta) \text{Wi} \mathbf{D}. \quad (3.15)$$

The momentum part of the EVSS formulation of (2.66)-(2.68) is similar to the original momentum equation (2.67), but now independent of β and therefore insensitive to the value. In numerical discretizations using this approach, especially in Galerkin methods, there are still numerical instabilities arising in oscillations in the solution due to the discretization of the second order derivatives of \mathbf{u} . To circumvent this, Rajagopalan et al. (1990) treats the derivatives of the strain rate tensor $\mathbf{D} = \frac{1}{2}(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)$ differently by first interpolating the components of \mathbf{D} and then evaluating the derivatives of it. This leads to an additional equation such that the discretization reads: find $(\mathbf{u}_h, p_h, \boldsymbol{\Sigma}_h, \mathbf{D}_h) \in \mathcal{V} \times \mathcal{Q} \times \mathcal{S} \times \mathcal{D}$ such that

$$\begin{aligned} \int_{\Omega} \nabla \cdot \mathbf{u}_h q_h \, dV &= 0, \quad \forall q_h \in \mathcal{Q}, \\ \int_{\Omega} \text{Re}(\nabla \mathbf{u}_h)^T : \nabla \mathbf{v}_h \, dV + \int_{\Omega} \boldsymbol{\Sigma}_h : \nabla \mathbf{v}_h \, dV - \int_{\Omega} \nabla \cdot \mathbf{v}_h p_h \, dV &= 0, \quad \forall \mathbf{v}_h \in \mathcal{V}, \\ \int_{\Omega} \left(\boldsymbol{\Sigma}_h + \text{Wi} \overset{\nabla}{\boldsymbol{\Sigma}}_h - (1 - \beta) \text{Wi} \overset{\nabla}{\mathbf{D}}_h \right) : \boldsymbol{\sigma}_h \, dV &= 0, \quad \forall \boldsymbol{\sigma}_h \in \mathcal{S}, \\ \int_{\Omega} (\mathbf{D}_h - \nabla \mathbf{u}_h - (\nabla \mathbf{u}_h)^T) : \boldsymbol{\delta}_h \, dV &= 0, \quad \forall \boldsymbol{\delta}_h \in \mathcal{D}. \end{aligned} \quad (3.16)$$

Sun et al. (1996) presented an adapted version of the EVSS called the Adaptive Viscoelastic Stress Splitting (AVSS) method in which the splitting of the polymeric stress is conducted as follows:

$$\boldsymbol{\Sigma} := \boldsymbol{\tau} - \zeta_A \mathbf{D} \quad (3.17)$$

with the aim of ζ_A being allowed to change adaptively such that the viscous contribution to $\boldsymbol{\Sigma}$ is at least of the same order as that of the elastic contribution. Hereby, ζ_A should be chosen as:

$$\zeta_A := a \frac{h |\boldsymbol{\tau}_h|_{\max}}{|\mathbf{u}_h|_{\max}} \quad (3.18)$$

with the coefficient $a = \mathcal{O}(1)$. In this case the viscous and elastic terms are balanced such that the kinematics would not be too sensitive to changes in the elastic stress.

The most common used derived method from the EVSS is the so called Discrete Elastic Viscous Stress Splitting (DEVSS, Gu enette and Fortin, 1995). In this case, equations (3.16) are

modified in such a way that no change of variables is required. Therefore, the decomposition of the polymeric stress reads as follows:

$$\boldsymbol{\Sigma} := \boldsymbol{\tau} - \zeta_D \mathbf{D}_h \quad (3.19)$$

where ζ_D in the simplest case is chosen $\zeta_D = (1 - \beta)$. Taking the divergence of (3.19) and inserting both sides into the discretized system (3.16) leads to the new system: find $(\mathbf{u}_h, p_h, \boldsymbol{\Sigma}_h, \mathbf{D}_h) \in \mathcal{V} \times \mathcal{Q} \times \mathcal{S} \times \mathcal{D}$ such that

$$\begin{aligned} \int_{\Omega} \nabla \cdot \mathbf{u}_h q_h \, dV &= 0, \quad \forall q_h \in \mathcal{Q}, \\ \int_{\Omega} \zeta_D (\mathbf{D}(\mathbf{u}_h) - \mathbf{D}_h) : \nabla \mathbf{v}_h \, dV + \int_{\Omega} \boldsymbol{\Sigma}_h : \nabla \mathbf{v}_h \, dV - \int_{\Omega} \nabla \cdot \mathbf{v}_h p_h \, dV &= 0, \quad \forall \mathbf{v}_h \in \mathcal{V}, \\ \int_{\Omega} \left(\boldsymbol{\Sigma}_h + \text{Wi} \overset{\nabla}{\boldsymbol{\Sigma}}_h - (1 - \beta) \text{Wi} \mathbf{D}(\mathbf{u}_h) \right) : \boldsymbol{\sigma}_h \, dV &= 0, \quad \forall \boldsymbol{\sigma}_h \in \mathcal{S}, \\ \int_{\Omega} (\mathbf{D}_h - \mathbf{D}(\mathbf{u}_h)) : \boldsymbol{\delta}_h \, dV &= 0, \quad \forall \boldsymbol{\delta}_h \in \mathcal{D}. \end{aligned} \quad (3.20)$$

It is obvious that in the continuous setting this adaption does not make any sense. However, for the discretized system, where \mathbf{D}_h and $\mathbf{D}(\mathbf{u}_h)$ have different representations, it has the advantage that there is no need of a change of variable in the constitutive equation and no derivative of the rate-of-strain tensor is required (Owens and Phillips, 2002).

3.3.3. Log-Conformation Formulation

A newer idea for stabilizing numerical simulations of viscoelastic flow is treating other instabilities than the ones arising from change of type to hyperbolic behaviour. Fattal and Kupferman (2004) found that for increasing Weissenberg numbers the polymeric stresses are experiencing a combination of deformation and convection leading to exponential profiles with rising steepness. These are obviously poorly approximated from numerical schemes which usually use polynomial basis functions. Therefore, there are high restrictions to the mesh size h of the triangulation. This means that for a proper approximation of the stresses either exponential basis functions are needed or a change of variables to new variables that scale logarithmically with the stresses must be accomplished.

In both cases the procedures require the stress field to remain strictly positive which can be achieved by using the positive definite conformation tensor $\mathbf{c}(\mathbf{x}, t)$ of the stresses defined in equations (2.45)-(2.46). The conformation tensor has a well-defined matrix logarithm read (Fattal and Kupferman, 2005):

$$\boldsymbol{\Psi}(\mathbf{x}, t) := \log \mathbf{c}(\mathbf{x}, t). \quad (3.21)$$

This logarithmic transformation should remove instabilities due to exponential grow in the stresses.

At first, $\boldsymbol{\tau}(\mathbf{x}, t)$ needs to be defined in terms of the conformation tensor, in this case for an Oldroyd B model (Fattal and Kupferman, 2005):

$$\boldsymbol{\tau} := \frac{1}{\text{Wi}} (\mathbf{c} - \mathbf{I}). \quad (3.22)$$

The constitutive equation for the conformation tensor conducted from (2.68) is:

$$\frac{\partial \mathbf{c}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{c} - (\nabla \mathbf{u}) \mathbf{c} - \mathbf{c} (\nabla \mathbf{u})^T = \frac{1}{Wi} (\mathbf{I} - \mathbf{c}). \quad (3.23)$$

To build the matrix-logarithm Ψ of \mathbf{c} the deformation terms in equation (3.23) including $\nabla \mathbf{u}$ need to be decomposed into a pure extensional and a pure rotational part:

$$\nabla \mathbf{u} = \mathbf{W} + \mathbf{D} + \mathbf{N} \mathbf{c}^{-1} \quad (3.24)$$

where \mathbf{W} and \mathbf{N} are skew-symmetric and \mathbf{D} is symmetric. Substituting (3.24) into (3.23) we obtain:

$$\frac{\partial \mathbf{c}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{c} - (\mathbf{W} \mathbf{c} - \mathbf{c} \mathbf{W}) - 2\mathbf{D} \mathbf{c} = \frac{1}{Wi} (\mathbf{I} - \mathbf{c}) \quad (3.25)$$

where \mathbf{D} generates a pure area preserving extension and \mathbf{W} generates a pure rotation. Now we can derive the equation for Ψ from equation (3.25) (Fattal and Kupferman, 2005):

$$\frac{\partial \Psi}{\partial t} + (\mathbf{u} \cdot \nabla) \Psi - (\mathbf{W} \Psi - \Psi \mathbf{W}) - 2\mathbf{D} = \frac{1}{Wi} e^{-\Psi} (\mathbf{I} - e^{\Psi}). \quad (3.26)$$

This equation can now be used instead of the constitutive equation for the Oldroyd B model (2.68) for discretization.

3.4. Stabilization Methods for the Constitutive Equations

Whereas in the previous section stabilization methods for handling the mixed hyperbolic-elliptic-parabolic nature of the system for discretization were presented, we now have a closer look on the handling of the convection term in the convection-dominated hyperbolic convection-diffusion equation for the constitutive material (Eq. 2.68). As mentioned in Sec. 3.2, a kind of upwinding strategy comes into play for the discretization of the convective term. Since the FEM is the most common method or basis for combination with different methods, we want to present the most common upwinding strategies for the weak formulation (3.6) of the constitutive equation (2.68) at this point .

The two kinds of stabilization methods are used in various combinations though the Discrete Elastic Viscous Stress Splitting (DEVSS) together with the Streamline Upwinding Petrov-Galerkin (SUPG) or DG are the most common. The combinations are denoted in literature by using a slash, e.g. DEVSS/DG or DEVSS/SUPG (Claus and Phillips, 2013; Kim et al., 2004).

3.4.1. Streamline Upwinding (Petrov-Galerkin) Methods

The SUPG was introduced by Brooks and Hughes (1982). In this method the weak formulation of the constitutive equation is replaced by: find $\tau \in \mathcal{S}$ such that:

$$\int_{\Omega} \left(\tau + Wi \tau^{\nabla} - (1 - \beta) \mathbf{D} \right) : (\sigma + \alpha \mathbf{u} \cdot \nabla \sigma) dV = 0, \quad \forall \sigma \in \mathcal{S}. \quad (3.27)$$

Here, the test function was changed to an upwinding formulation containing the convection of the test function. The upwinding parameter α should be of order $\mathcal{O}(\frac{h}{U})$ where for the

characteristic U different choices are possible, e.g. it can be the magnitude of the velocity at the center of biquadratic quadrilateral elements in a FEM discretization, or a characteristic velocity of the flow, e.g. the one used for non-dimensionalization of the equations u_c (Marchal and Crochet, 1987; Owens and Phillips, 2002). In case of spectral methods, a version of the SUPG can be used with a different α . For spectral elements methods the stabilization by using SUPG is more difficult since the high-order approach is of highly oscillatory nature (Owens and Phillips, 2002).

It is also possible to use a Streamline Upwinding (SU) method where the upwinding is only applied to the convective term itself such that there is an additional artificial diffusion in the system. This leads to a more stable formulation with a loss of accuracy (Owens and Phillips, 2002): find $\boldsymbol{\tau} \in \mathcal{S} \forall \boldsymbol{\sigma} \in \mathcal{S}$ such that:

$$\int_{\Omega} \left(\boldsymbol{\tau} + \text{Wi} \frac{\nabla}{U} \boldsymbol{\tau} \right) : \boldsymbol{\sigma} \, dV + \int_{\Omega} \text{Wi} (\mathbf{u} \cdot \nabla \boldsymbol{\tau}) : (\alpha \mathbf{u} \cdot \nabla \boldsymbol{\sigma}) \, dV = \int_{\Omega} (1 - \beta) \mathbf{D} : \boldsymbol{\sigma} \, dV. \quad (3.28)$$

SUPG is for example used in newer times by Kim et al. (2004), Coronado et al. (2007), Varchanis et al. (2019).

3.4.2. Discontinuous Galerkin Formulation with Upwinding

The DG upwinding method used in the context with FEM was developed by the group around Fortin (Fortin and Fortin, 1989; Fortin and Fortin, 1990; Fortin and Zine, 1992). In this DG approach the steps described in the following Chap. 4 are followed for the convective term of the constitutive equation introducing weighted fluxes for the inflow and outflow flux of an element. So the weak formulation reads: find $\boldsymbol{\tau} \in \mathcal{S} \forall \boldsymbol{\sigma} \in \mathcal{S}$ such that:

$$\int_{\Omega} \left(\boldsymbol{\tau} + \text{Wi} \frac{\nabla}{U} \boldsymbol{\tau} \right) : \boldsymbol{\sigma} \, dV - \oint_{\Gamma_I} \text{Wi} \{ \mathbf{u} \} \cdot \mathbf{n} \llbracket \boldsymbol{\tau} \rrbracket : (f_2 \boldsymbol{\sigma}^+ - f_1 \boldsymbol{\sigma}^-) = \int_{\Omega} (1 - \beta) \mathbf{D} : \boldsymbol{\sigma} \, dV, \quad (3.29)$$

with the following functions depending on the upwind parameter $0 \leq \alpha \leq 1$:

$$f_1(\alpha) := \begin{cases} \alpha, & \text{if } \{ \mathbf{u} \} \cdot \mathbf{n} < 0 \\ 1 - \alpha, & \text{if } \{ \mathbf{u} \} \cdot \mathbf{n} \geq 0 \end{cases} \quad (3.30)$$

$$f_2(\alpha) := \begin{cases} \alpha, & \text{if } \{ \mathbf{u} \} \cdot \mathbf{n} < 0 \\ 1 - \alpha, & \text{if } \{ \mathbf{u} \} \cdot \mathbf{n} \geq 0 \end{cases}$$

Most recent work using DG are e.g. Hulsen et al. (2005), Claus and Phillips (2013) or Kynch and Phillips (2017).

4. The Discontinuous Galerkin Discretization for Viscoelastic Single-Phase Flow

In this chapter we give an overview over the DG method and present the discretization for viscoelastic single-phase flow. It should not give a general description of the methods but rather introduce the methods used to discretize the system of equations (2.69)-(2.71), followed by the last section where these methods are brought together in the new discretization of the system. For a more detailed introduction, the reader is referred to the papers of Cockburn (e.g. Cockburn et al., 2000; Cockburn et al., 2005b) and the books by Li (2006), Hesthaven and Warburton (2008) and Di Pietro and Ern (2012). The descriptions in this chapter mainly rely on the textbook of Hesthaven and Warburton (2008) and the work of Cockburn et al. on LDG methods (esp. Castillo et al., 2000; Cockburn et al., 2002). For the multi-phase application the reader is referred to the next Chap. 5.

4.1. State of Knowledge

The DG method was first proposed by Reed and Hill (1973) to discretize the transport equation of neutrons and the scheme was further analyzed for this purpose by Lesaint and Raviart (1974). Arnold (1982) presented an interior penalty method for inner edges between cells in DG, originally developed by Babuška (1973) to weakly impose Dirichlet BC.

Cockburn and Shu extended the DG method to hyperbolic (1989) and general (1991a) conservation laws and the Euler equations (1991b) with a Runge-Kutta-Scheme.

Arnold et al. (2002) presented a DG discretization for elliptic problems. Detailed work about the stability of the DG method for the Stokes and Navier-Stokes System can be found in Girault et al. (2005). A good overview of the research done in the field of DG methods until 2005 gives the book of Cockburn et al. (2005b) as editors with many invited papers of researchers dealing with DG methods in different applications.

Shahbazi et al. (2007) showed high-order numerical solutions to the Navier-Stokes equations using a local Lax-Friedrichs flux for the convective terms and a symmetric interior penalty method with an explicit penalty parameter presented by Shahbazi (2005) for the Poisson operator. In 2007 Cockburn et al. presented a divergence free velocity approximation of the Navier-stokes equations and Cockburn et al. (2009) showed an equal order approach for the Navier-Stokes system using the penalty method developed by Arnold (1982) as a basis.

In recent years the focus in DG methods has been on space-time DG methods (e.g. Klaij et al., 2006; Rhebergen et al., 2013), also in the context of XDG (e.g. Lehrenfeld, 2015a; Lehrenfeld, 2015b), and Hybridizable Discontinuous Galerkin (HDG) where a super-convergence with

$k + 2$ can be shown for the velocity (Cesmelioglu et al., 2017).

Since in the DG method the cells are only coupled over fluxes with their neighbouring cells the communication between processors for parallel computing are kept by a minimum. So the DG method scales up to more than 10^3 cores as shown by Sonntag and Munz (2017).

Local Discontinuous Galerkin Method

In combination with the previously mentioned Runge-Kutta-Scheme Cockburn and Shu (1998) developed a LDG method dividing the elliptic Poisson operator into two first-order equations building a hyperbolic system which is stabilized with penalized fluxes. They expanded their scheme to convection-dominated convection-diffusion problems (Cockburn and Shu, 2001). In the following years the LDG method was extended to the Stokes system (Cockburn et al., 2002; Cockburn et al., 2005b) and the Navier-Stokes system (Cockburn et al., 2005a). The LDG method can be used for a wide range of problems, since it has a high degree of locality and is locally conservative, especially for the use in convection-diffusion problems. It can easily handle elements of arbitrary shape including hanging nodes and can therefore be used for adaptive mesh refinement also in elliptic problems (Castillo et al., 2000).

The Use of the Discontinuous Galerkin Method in the Viscoelastic Case

As already mentioned in Sec. 3.4.2, parallel to the development of the DG method for Newtonian fluid applications Fortin and Fortin (1989) started to use the DG method in the viscoelastic field. Up to then, viscoelastic fluid flow was completely discretized using FEM in combination with a SU for the convective terms in the constitutive equations. Fortin and Fortin (1989) presented the idea to discretize the stresses using the DG approach of Lesaint and Raviart (1974). They further developed their ansatz in the following years by utilizing a decoupled Generalized Minimal Residual (GMRES) scheme (e.g. Fortin and Fortin, 1989; Fortin and Fortin, 1990; Fortin and Zine, 1992; Fortin et al., 1992; Gu nette and Fortin, 1995) and introduced the EVSS method first presented by Rajagopalan et al. (1990).

The group around Baaijens (1994a) published a series of works around the Phan-Thien Tanner (PTT) model for different benchmark problems using a low order DG method for discretization of the stresses (Baaijens, 1994a; Baaijens, 1994b) and compared their results with physical experiments (e.g. Baaijens et al., 1997). They also investigated the UCM model (Baaijens, 1998). An overview over the work done up to the turn of the century can be found in Bogaerds et al. (2000) and Fortin et al. (2000).

Afterwards, the DG scheme became a standard method discretizing the stresses, namely the convective terms, in the constitutive equations and is widely used in many numerical schemes (e.g. Hulsen et al., 2005; Kim et al., 2005; Claus and Phillips, 2013).

Of particular interest for this work is one publication of Mirzakhali and Nejat (2015). This is the only work on an exclusively high order DG scheme for Oldroyd B fluid flow. However, they use a highly decoupled algorithm for simulation of the 4 : 1 contraction benchmark problem. Further publications on exclusively high-order DG schemes are unknown to the author.

4.2. Definitions for the Discontinuous Galerkin Space

The definitions of this section are mainly derived from Kummer et al. (2020b) and the book of Hesthaven and Warburton (2008). The definitions at this point are for single-phase flow and are extended to the multi-phase purpose in Sec. 5.2.

- We have a physical domain $\Omega \subset \mathbb{R}^2$ with a boundary $\partial\Omega$ and we have a corresponding computational domain $\Omega_h \subset \mathbb{R}^2$ with its boundary $\delta\Omega_h$ which must be polygonal and simply connected.
- On Ω_h we have a space filling triangulation as a numerical grid with geometry-conforming non-overlapping elements $\mathfrak{K} := \{K_1, \dots, K_N\}$ with a characteristic mesh size h .
- The physical domain Ω is approximated by the union of all elements:

$$\Omega \approx \Omega_h := \bigcup_{i=1}^N K_i. \quad (4.1)$$

- $\Gamma := \bigcup_j \partial K_j$ is the union of all edges or the skeleton of the grid and $\Gamma_I := \Gamma \setminus \delta\Omega_h$ is the union of all interior edges. Furthermore, we define $\Gamma_D := \partial\Omega_{h,D}$, $\Gamma_N := \partial\Omega_{h,N}$, $\Gamma_S := \partial\Omega_{h,S}$ for Dirichlet, Neumann, and free-slip boundaries on the computational grid Ω_h , respectively.
- We denote a normal field \mathbf{n}_Γ , on $\delta\Omega_h$ it defines an outer normal.
- The superscript ψ^- defines the information in the interior of an element while the superscript ψ^+ defines the exterior information of the neighbouring cell for a field $\psi \in C^0(\Omega_h \setminus \Gamma_I)$ with:

$$\psi^- := \lim_{\varepsilon \searrow 0} \psi(\mathbf{x} - \varepsilon \mathbf{n}_\Gamma) \quad \text{for } \mathbf{x} \in \Gamma, \quad (4.2)$$

$$\psi^+ := \lim_{\varepsilon \searrow 0} \psi(\mathbf{x} + \varepsilon \mathbf{n}_\Gamma) \quad \text{for } \mathbf{x} \in \Gamma_I. \quad (4.3)$$

- It follows the definition that \mathbf{n}_Γ^+ points outwards into the neighbouring cell, whereas \mathbf{n}_Γ^- points inwards and therefore, $\mathbf{n}_\Gamma^- = -\mathbf{n}_\Gamma^+$.
- The jump and mean value of a component of a tensor of arbitrary order on inner edges Γ_I are, respectively:

$$[[\psi]] := \psi^- - \psi^+, \quad (4.4)$$

$$\{\psi\} := \frac{1}{2} (\psi^- + \psi^+). \quad (4.5)$$

- The jump and mean value on boundary edges $\delta\Omega_h$ are:

$$[[\psi]] := \psi^-, \quad (4.6)$$

$$\{\psi\} := \psi^-. \quad (4.7)$$

- We define the broken polynomial space of total degree k as:

$$\mathbb{P}_k(\mathfrak{K}) := \{f \in L^2(\Omega_h); \forall K \in \mathfrak{K} : f|_K \text{ is polynomial and } \deg(f|_K) \leq k\}. \quad (4.8)$$

Note that the term broken refers to the piecewise polynomial space with jumps on cell boundaries.

- We define the function space for test and trial functions for D_v dependent variables with discretization of different polynomial degree k . Therefore, we define the degree-vector $\mathbf{k} := (k_1, \dots, k_{D_v})$ and get:

$$\mathbb{V}_{\mathbf{k}} := \prod_{i=1}^{D_v} \mathbb{P}_{k_i}(\mathfrak{K}) \quad (4.9)$$

- We define for $u_K, v_K \in \mathbb{V}_{\mathbf{k}}$ a local inner product and a local L^2 -norm for a cell K :

$$(u_K, v_K)_K := \int_K u_K v_K \, dx, \quad \|u_K\|_K^2 := (u_K, u_K)_K \quad (4.10)$$

- and for $u_h, v_h \in \mathbb{V}_{\mathbf{k}}$ a global broken inner product and a global broken norm:

$$(u_h, v_h)_{\Omega_h} := \sum_{i=1}^N (u_h, v_h)_K, \quad \|u_h\|_{\Omega_h}^2 := (u_h, u_h)_{\Omega_h}. \quad (4.11)$$

- For $\mathbf{u}_h \in C^1(\Omega_h)$ the broken gradient is defined by $\nabla_h \mathbf{u}_h$ on the domain Ω_h where differentiation at the jumps on Γ is excluded and analogously, $\nabla_h \cdot \mathbf{u}_h$ is the broken divergence.

4.3. Introduction to the Discontinuous Galerkin Discretization

In this section we present the DG method exemplarily on a single-phase non-linear conservation equation for a scalar quantity u of the form:

$$\frac{\partial u}{\partial t} + \nabla \cdot f(u) = 0, \quad u \in \Omega \quad (4.12)$$

using the assumptions and definitions made in section 4.2. We generate the weak formulation of the problem by multiplying the eq. (4.12) with a test function v and performing integration by parts:

$$\int_{\Omega} \frac{\partial u}{\partial t} v \, dV + \int_{\Omega} f(u) \cdot \nabla v \, dV = - \int_{\partial\Omega} f(u) \cdot \mathbf{n}_{\Gamma} \, dS. \quad (4.13)$$

This weak formulation of the problem is used in many different discretizations beside the DG method such as FEM. For the DG discretization of this example we use the Method of Lines (MOL), meaning that we concentrate on the spatial discretization of the problem excluding the time, since we have a parabolic initial boundary value problem of first order (Schiesser, 2012). The time discretization follows in Sec. 4.5.

We now approximate the domain Ω with a grid \mathfrak{K} of N non-overlapping elements K_j and on each element the solution is defined by a polynomial $u_K(\mathbf{x}, t) \in \mathbb{P}_k(\mathfrak{K})$ of order k :

$$u_K(\mathbf{x}, t) := \sum_{i=0}^{N_k} \hat{u}_i(t) \phi_i(\mathbf{x}) \quad (4.14)$$

with N_k being the degrees of freedom (DOF) for the polynomial of order k with respect to the dimension of the problem. We choose a modal form with a local polynomial basis for the DG ansatz, e.g. $\tilde{\phi}_i(\mathbf{x}) = x^i$. Additionally, we demand the basis vectors to be orthogonal in applying the Gram-Schmidt process to them such that

$$\int_K \phi_i \phi_j \, dV = \delta_{ij}. \quad (4.15)$$

By summing up the local polynomials we derive a global solution $u_h(\mathbf{x}, t) \in \mathbb{V}_k$ which is approximated by piecewise polynomials of order k :

$$u(\mathbf{x}, t) \approx u_h(\mathbf{x}, t) := \bigoplus_{j=1}^N u_{K_j}(\mathbf{x}, t). \quad (4.16)$$

Obviously, when we insert this approximated global solution into the original problem (4.12) we obtain a residual:

$$R_h(\mathbf{x}, t) = \frac{\partial u_h}{\partial t} + \nabla \cdot f(u_h). \quad (4.17)$$

We now define a space of test functions $v_K(\mathbf{x}, t) \in \mathbb{P}_{kk}(\mathfrak{K})$ with the same basis ϕ like the trial function $u_K(\mathbf{x}, t)$ (Galerkin approach). Since the basis is orthonormal, the coefficients for the test function become one such that:

$$v_K(\mathbf{x}, t) := \sum_{i=0}^k \phi_i(\mathbf{x}) \quad (4.18)$$

and the global discrete test function for $v_h(\mathbf{x}, t) \in \mathbb{V}_k$ reads:

$$v(\mathbf{x}, t) = v_h(\mathbf{x}, t) := \bigoplus_{j=1}^N v_{K_j}(\mathbf{x}, t). \quad (4.19)$$

Since we want to minimize the residual R_h for the discretized weak formulation of the problem (4.13), we require it to be orthogonal to all test functions which means:

$$\int_{\Omega_h} R_h(\mathbf{x}, t) \phi_i(\mathbf{x}) \, dV = 0 \quad \text{for } 0 \leq i \leq k \quad (4.20)$$

on all elements. If we now insert the approximated test function $v_h(\mathbf{x}, t) \in \mathbb{V}_k$ (Eq. (4.19)) and the trial function $u_h(\mathbf{x}, t) \in \mathbb{V}_k$ (Eq. 4.16) into Eq. (4.13), we get a minimization problem for the semi-discrete scheme: find $u_h(\mathbf{x}, t) \in \mathbb{V}_k \forall v_h(\mathbf{x}, t) \in \mathbb{V}_k$ such that:

$$\int_{\Omega_h} \frac{\partial u_h}{\partial t} v_h \, dV - \int_{\Omega_h} f_h(u_h) \nabla_h v_h \, dV = - \int_{\Gamma} f^*(u_h^-, u_h^+, \mathbf{n}_\Gamma) \llbracket v_h \rrbracket \, dV. \quad (4.21)$$

The solution on the edges between elements is now multiple defined and needs to be connected using flux functions, here denoted as $f^*(u^-, u^+, \mathbf{n}_\Gamma)$. Therefore, we need to choose, which solution u_h^- or u_h^+ or a combination of both is correct. We call this a numerical flux. The numerical flux plays an important role to guarantee stability of the formulation (4.21) by representing the flow of information in the underlying conservation law (Eq. 4.12). Additionally, it must be consistent with the original equation (4.12), e.g. keeping it single-valued if originally so, it must enforce the BC on $\partial\Omega$ correctly, and it must be chosen such that it reduces to a monotone scheme in the low order limit which means that it then displays a monotone finite volume approximation. This is ensured by the flux being non-decreasing in the first argument and non-increasing in the second argument.

We now define $f(u) := au(\mathbf{x}, t)$ and therefore, let eq. (4.12) be linear. We then can define the flux as:

$$f^*(u_h^-, u_h^+, \mathbf{n}_\Gamma) := \mathbf{n}_\Gamma \{ \alpha u_h \} + |\alpha| \frac{1-\alpha}{2} \llbracket u_h \rrbracket \quad (4.22)$$

with $0 \leq \alpha \leq 1$ fulfilling the properties above. This flux displays in the limit $\alpha = 1$ a central flux and in the limit $\alpha = 0$ an upwinding flux taking the information along the streamline from where it is coming. While the upwind formulation generates less jumps between the elements than the central flux, it is also more dissipative than the latter.

In a second example we assume eq. (4.12) to be non-linear. In this case, we can choose the more general Lax-Friedrichs flux, which includes the special cases central and streamline upwinding flux:

$$f^{*LF}(u_h^-, u_h^+, \mathbf{n}_\Gamma) := \frac{f(u_h^-) + f(u_h^+)}{2} + \frac{C^{LF}}{2} \mathbf{n}_\Gamma \cdot \llbracket u_h \rrbracket \quad (4.23)$$

with different definitions for C^{LF} . In the case of the local Lax-Friedrichs flux it is defined as:

$$C^{LF} \geq \max_{\min(u_h^-, u_h^+) \leq w \leq \max(u_h^-, u_h^+)} |f_u(w)| \quad (4.24)$$

with $f_u = \frac{\partial f}{\partial u}$ being the Jacobian of the flux. The global Lax-Friedrichs flux, which is more dissipative than the local Lax-Friedrichs flux, is defined as:

$$C^{LF} \geq \max_{\inf(u_h(\mathbf{x})) \leq w \leq \sup(u_h(\mathbf{x}))} |f_u(w)|. \quad (4.25)$$

If we consider the weak formulation in Eq. (4.21) for all governing equations including appropriate problem-dependent fluxes such as e.g. the Lax-Friedrichs flux, we have a system of $D_v \times k$ equations for the same amount of unknown coefficients of the form:

$$\mathcal{M}_M \frac{d}{dt} \mathbf{u}_h - \mathcal{M}_S^T \mathbf{f}_h(\mathbf{u}_h) = \mathbf{b} \quad (4.26)$$

with $\mathcal{M}_{Mij} = (\phi_i, \phi_j)$ is the mass matrix of the system (4.21) and $\mathcal{M}_{Sij} = (\phi_i, \nabla \phi_j)$ is the stiffness matrix of the system (4.21). So a system of the following structure has to be solved for a specific time level t :

$$A_U(\mathbf{U}) \mathbf{U} = \mathbf{b} \quad (4.27)$$

which simplifies to $A_U \mathbf{U} = \mathbf{b}$ for the linear case. The methods for solving these non-linear and linear systems will be described in Chap. 6.

4.4. The Local Discontinuous Galerkin Method

The following section is mainly extracted from Castillo et al., 2000. We consider an elliptic Poisson problem of the form:

$$\begin{aligned} -\Delta u &= f && \text{in } \Omega, \\ u &= u_D && \text{on } \partial\Omega_D, \\ \frac{\partial u}{\partial \mathbf{n}} &= \mathbf{u}_N \cdot \mathbf{n} && \text{on } \partial\Omega_N. \end{aligned} \quad (4.28)$$

We rewrite this problem to a system of first order equations by introducing an auxiliary variable $\mathbf{g} := \nabla u$:

$$\begin{aligned} \mathbf{g} &= \nabla u && \text{in } \Omega, \\ -\nabla \mathbf{g} &= f && \text{in } \Omega, \\ u &= u_D && \text{on } \partial\Omega_D, \\ \frac{\partial u}{\partial \mathbf{n}} &= \mathbf{u}_N \cdot \mathbf{n} && \text{on } \partial\Omega_N. \end{aligned} \quad (4.29)$$

We use a space filling triangulation as defined in Sec. 4.2 with a grid of geometry conforming non-overlapping elements K . The weak formulation of Eq. (4.29) is found by multiplying the equations with arbitrary smooth test functions, performing integration by parts, by summing over all elements in \mathfrak{K} . We get the global formulated problem: find $(u_h, \mathbf{g}_h) \in \mathbb{V}_{\mathbf{k}}$, such that

$$\begin{aligned} \int_{\Omega_h} \mathbf{g}_h \cdot \mathbf{q}_h \, dV &= - \int_{\Omega_h} u_h \nabla_h \cdot \mathbf{q}_h \, dV + \int_{\Gamma} f^*_u \llbracket \mathbf{q}_h \rrbracket \cdot \mathbf{n}_{\Gamma} \, dS, \\ \int_{\Omega_h} \mathbf{g}_h \cdot \nabla_h v_h \, dV &= \int_{\Omega_h} f v_h \, dV + \int_{\Gamma} f^*_g \llbracket v_h \rrbracket \cdot \mathbf{n}_{\Gamma} \, dS \quad \forall (v_h, \mathbf{q}_h) \in \mathbb{V}_{\mathbf{k}} \end{aligned} \quad (4.30)$$

with the numerical fluxes:

$$f^*_u := \{u_h\} - C_{12} \cdot \llbracket u_h \rrbracket - C_{22} \llbracket \mathbf{g}_h \rrbracket, \quad (4.31)$$

$$f^*_g := \{\mathbf{g}_h\} - C_{11} \llbracket u_h \rrbracket - C_{12} \llbracket \mathbf{g}_h \rrbracket \quad (4.32)$$

and on the boundary:

$$f^*_u := \begin{cases} u_D & \text{on } \Gamma_D, \\ u_h^+ - C_{22}(\mathbf{g}_h^+ \cdot \mathbf{n} - \mathbf{u}_n \cdot \mathbf{n}) & \text{on } \Gamma_N, \end{cases} \quad (4.33)$$

$$f^*_g := \begin{cases} \mathbf{g}_h^+ - C_{11}(u_h^+ \mathbf{n} - u_D \mathbf{n}) & \text{on } \Gamma_D, \\ \mathbf{u}_N & \text{on } \Gamma_N. \end{cases} \quad (4.34)$$

The LDG method is now obtained, if we choose $C_{22} = 0$ because then the numerical flux f^*_u does not depend on \mathbf{g}_h . Hence, the auxiliary variable \mathbf{g} can now be solved locally in terms of u_h . So the final problem has still only u as an unknown variable. This property of being locally solvable gives the name to the LDG method. Remark that the result is also independent of the coefficient vector C_{12} .

The existence of a solution is guaranteed, if for all $v_h \in \mathbb{V}_{\mathbf{k}}$

$$\int_{\Omega_h} \nabla_h \cdot \mathbf{q}_h \, dV = 0 \quad \forall \mathbf{q}_h \in \mathbb{V}_{\mathbf{k}}. \quad (4.35)$$

This is easily satisfied because $(\nabla_h \mathbb{V}_k) \subset \mathbb{V}_k$ since \mathbb{V}_k is defined in the broken polynomial space $\mathbb{P}_k(\mathcal{K})$ of total degree k (Eqs. (4.8) and (4.9)). This means that the LBB condition holds for the LDG discretization.

The method is conservative since Eqs. (4.29) are weakly enforced on an element-by-element basis using numerical fluxes between elements. These numerical fluxes enhance the stability of the discretization and therefore, enhance the quality of the approximation by penalizing the jumps between element edges.

If the stabilization coefficient C_{11} is chosen of order one and polynomials of at least degree k are used in all elements, the rate of convergence in the L^2 -Norm of u and \mathbf{g} are of order $k + \frac{1}{2}$ and k , respectively. If the stabilization parameter is chosen to be of order h^{-1} ($\mathcal{O}(h^{-1})$) an order of convergence of $k + 1$ and k can be reached. Since we take the dependency on h into account, we expect a convergence order of $k + 1$ for the velocities and k for the stresses in our governing system of equations. Hence, the constants in Eqs. (4.32) and (4.34) are defined as follows:

$$C_{11} := \frac{\gamma_3}{h_{\min}}, \quad C_{12} := \mathbf{0}. \quad (4.36)$$

with γ_3 being a user defined penalty value. Note that the purpose of the coefficient C_{11} is to ensure stability whereas the parameter C_{12} only reduces the sparsity of the matrix and enhances the accuracy of the method. Thus, we have set it to zero following Cockburn et al. (2002). In Sec. 7.1 a verifying study of the implementation of the LDG in the BoSSS framework by means of a convergence study for a non-polynomial manufactured solution following Cockburn et al. (2002) can be found.

4.5. Time Discretization

We derive the information of this section from Ferziger and Perić (2008) and Deville et al. (2002).

The time discretization is a parabolic problem where a force at a particular time only influences the development of future time-steps, contrary to the spatial discretization (elliptic problem where a force can influence the whole domain). Hence, we have for eq. (4.21) a typical initial value problem of the form:

$$\frac{du}{dt} = f(t, u(t)), \quad u(t_0) = u_0. \quad (4.37)$$

where $f(t, u(t))$ is usually non-linear, if convective terms are playing a role. We convert the problem to the autonomous problem, where the right-hand-side depends only on $u(t)$:

$$\frac{du}{dt} = f(u(t)), \quad u(t^0) = u^0. \quad (4.38)$$

We now search a solution $u(t^1) = u^1$ of the equation for $t^1 = t^0 + \Delta t$ and so on, which is formally the integral:

$$u(t) - u^0 = \int_{t^0}^t f(u(s)) ds \quad (4.39)$$

We approximate this integral using a numerical quadrature which is for a wide range of methods:

$$\int_{t^0}^t f(u(s)) \, ds \approx \sum_{i=1}^l w_i f(u(t^i)), \quad t^i \leq t \quad (4.40)$$

with w_i being the quadrature weights for a set of discrete time values t^i . Consequently, the numerical solution \tilde{u} reads:

$$\tilde{u} - u^0 = \sum_{i=1}^l w_i f(\tilde{u}(t^i)), \quad t^i \leq t. \quad (4.41)$$

Different approaches exist and they can be divided in two groups which are the explicit methods, and implicit methods. The simplest explicit method is the explicit Euler scheme $u^{n+1} = u^n + f(u^n) \Delta t$, and the simplest implicit method is the implicit Euler scheme $u^{n+1} = u^n + f(u^{n+1}) \Delta t$ with $u^n = u(t^n)$.

Whereas in case of implicit methods the whole system of equations needs to be solved, this is not necessary for explicit methods. But, although explicit methods are thus easier to implement and need less storage and computational time than implicit methods, they are not unconditionally stable for increasing time-steps Δt . They must fulfil the Courant-Friedrichs-Lewy (CFL) condition where $\Delta t < \frac{h}{u_c}$ with u_c being a characteristic velocity as defined in Sec. 2.5.2. This means that a fluid particle may not transit a full element during the time period Δt . For the implicit Euler method the time-step size can be arbitrary large. Thus, it is also used in this work for solutions of steady flow.

Whereas these methods are of first order, there are various time-stepping methods with higher orders. In this work we focus on implicit BDF schemes of the form:

$$\tilde{u}^{n+1} - \Delta t b_0 \tilde{q} f^{n+1} = \sum_{j=1}^z a_j \tilde{u}^{n+1-j} + \Delta t b_0 (1 - \tilde{q}) f^{n+1} \quad (4.42)$$

where z is the order of the BDF z scheme and $\tilde{q} = 1$. The values for the coefficients can be extracted from Tab. 4.1. Using eq. (4.42) we can also obtain the Crank-Nicholson (CN) scheme based on the trapezoidal rule by choosing $\tilde{q} = 0.5$.

Even though implicit methods are in general unconditionally stable, they are not necessarily A-stable. A-stability means that the following condition holds: If the scheme is applied to the Dahlquist equation:

$$\frac{du}{dt} = \lambda u(t) \quad \text{with } u(t_0) = 1, \quad (4.43)$$

with $\Re(\lambda) < 0$ for all real $\Delta t > 0$, it is A-stable if $\lim_{n \rightarrow \infty} u(n\Delta t) = 0$. For increasing z the region where this condition holds can shrink to a region with an opening angle θ (Fig. 4.1). Then the method is called A(θ)-stable. If this condition is satisfied, there is no unphysical growth, but it does not guarantee accuracy of the method, especially for stiff problems. In this work we use only methods up to BDF2 as they are always A-stable (Fig. 4.1).

4.6. Discretization of the Viscoelastic Governing Equations

If we want to discretize the governing equations (2.66) - (2.68), we need to define IC and BC on the boundary $\partial\Omega$ of a physical domain $\Omega \subset \mathbb{R}^2$ for the dependent variables in order to

Table 4.1.: Coefficients for the BDF z schemes for $z \leq 4$ and the Crank-Nicholson scheme (CN). The coefficients in Eq. (4.42) are related to the coefficients in the table as follows: $b_0 := \frac{\nu_0}{\nu}$, $a_j := \frac{\nu_j}{\nu}$ with $1 \leq j \leq 4$. IE is the implicit Euler method which is equal to the BDF1 scheme.

z	ν	ν_0	ν_1	ν_2	ν_3	ν_4	\tilde{q}
1(IE)	1	1	1				1
2	3	2	4	-1			1
3	11	6	18	-9	2		1
4	25	12	48	-36	16	-3	1
CN	1	1					0.5

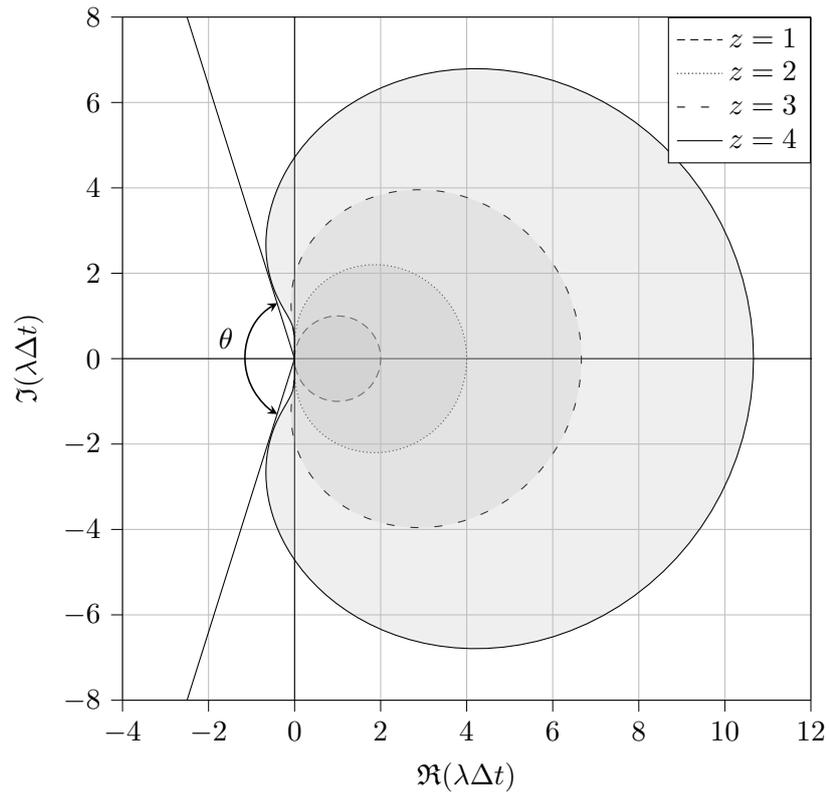


Figure 4.1.: A-stability regions of the BDF z schemes for $1 \leq z \leq 4$. The inner regions of the circles are unstable, the outer regions are stable. It can be seen that the schemes up to the BDF2 are always stable for all $\Re(\lambda\Delta t) < 0$. For the BDF4 the opening angle of the A(θ)-stability region is marked.

obtain a well-posed problem. The IC are:

$$\mathbf{u} = \mathbf{u}_0, \quad \boldsymbol{\tau} = \boldsymbol{\tau}_0 \quad \text{for } t = t_0. \quad (4.44)$$

On Dirichlet boundaries $\partial\Omega_D$ we define:

$$\mathbf{u} = \mathbf{u}_D \quad \text{on } \partial\Omega_D, \quad (4.45)$$

on boundaries with a condition of the Neumann type $\partial\Omega_N$ we have:

$$(-p\mathbf{I} + \beta(\nabla\mathbf{u} + (\nabla\mathbf{u})^T) + \boldsymbol{\tau}) \mathbf{n}_{\partial\Omega} = 0 \quad \text{on } \partial\Omega_N, \quad (4.46)$$

and we introduce a free-slip BC on $\partial\Omega_S$ with:

$$\begin{aligned} \mathbf{u} \cdot \mathbf{n}_{\partial\Omega} &= 0 \quad \text{and} \\ P_S (\beta(\nabla\mathbf{u} + (\nabla\mathbf{u})^T) + \boldsymbol{\tau}) \mathbf{n}_{\partial\Omega} &= 0 \quad \text{on } \partial\Omega_S, \end{aligned} \quad (4.47)$$

where $P_S = \mathbf{I} - \mathbf{n}_{\partial\Omega} \otimes \mathbf{n}_{\partial\Omega}$ denotes the orthogonal projection onto the boundary $\partial\Omega_S$.

4.6.1. Time Discretization using a Backward Difference Scheme

For the temporal discretization of Eq. (2.67) and Eq. (2.68) we use a BDF of second order (BDF2) since we aim for a higher order time discretization, but want to guarantee A-stability for all simulations. This leads to the following system

$$\begin{aligned} \nabla \cdot \mathbf{u}^{n+1} &= 0, \\ \text{Re} \left(\frac{3\mathbf{u}^{n+1}}{2\Delta t} + \mathbf{u}^{n+1} \cdot \nabla \mathbf{u}^{n+1} \right) \\ + \nabla p^{n+1} - \beta \Delta \mathbf{u}^{n+1} - \nabla \cdot \boldsymbol{\tau}^{n+1} &= \frac{\text{Re}}{\Delta t} \left(2\mathbf{u}^n - \frac{1}{2}\mathbf{u}^{n-1} \right), \\ \boldsymbol{\tau}^{n+1} + \text{Wi} \left(\frac{3\boldsymbol{\tau}^{n+1}}{2\Delta t} + \mathbf{u}^{n+1} \cdot \nabla \boldsymbol{\tau}^{n+1} - \nabla \mathbf{u}^{n+1} \cdot \boldsymbol{\tau}^{n+1} \right. \\ \left. - \boldsymbol{\tau}^{n+1} \cdot (\mathbf{u}^{n+1})^T \right) - (1 - \beta) \left(\nabla \mathbf{u}^{n+1} + (\nabla \mathbf{u}^{n+1})^T \right) &= \frac{\text{Wi}}{\Delta t} (2\boldsymbol{\tau}^n - \boldsymbol{\tau}^{n-1}). \end{aligned} \quad (4.48)$$

4.6.2. Spatial Discretization using Local Discontinuous Galerkin

First, to simplify the notation, let all properties in (4.48) which are related to times t^n and t^{n-1} (e.g. \mathbf{u}^n and \mathbf{u}^{n-1}), be absorbed in the right-hand sides. Since we need to satisfy the LBB condition for the Stokes system with $\beta = 1$, we use polynomials of degree k for velocity and stresses and of degree $k' = k - 1$ for the pressure (Babuška, 1973; Brezzi, 1974).

The DG discretization of the governing system of equations is the following: find $(\mathbf{u}_h, p_h, \boldsymbol{\tau}_h) \in \mathbb{V}_k$ such that for all $(\mathbf{v}_h, q_h, \boldsymbol{\sigma}_h = \boldsymbol{\sigma}_h^T) \in \mathbb{V}_k$:

$$\begin{aligned}
b(\mathbf{u}_h, q_h) + s_2(p_h, q_h) &= r_1(q_h), \\
\frac{3\text{Re}}{2\Delta t} (\mathbf{u}_h, \mathbf{v}_h) + c(\mathbf{u}_h, \mathbf{u}_h, \mathbf{v}_h) + b(p_h, \mathbf{v}_h) - a(\mathbf{u}_h, \mathbf{v}_h) \\
-d'(\mathbf{v}_h, \boldsymbol{\tau}_h) + s_1(\mathbf{u}_h, \mathbf{v}_h) &= r_2(\mathbf{v}_h), \quad (4.49) \\
\left(1 + \frac{3\text{Wi}}{2\Delta t}\right) (\boldsymbol{\tau}_h, \boldsymbol{\sigma}_h) + f(\mathbf{u}_h, \boldsymbol{\tau}_h, \boldsymbol{\sigma}_h) - g(\mathbf{u}_h, \boldsymbol{\tau}_h, \boldsymbol{\sigma}_h) - g(\mathbf{u}_h, \boldsymbol{\tau}_h, \boldsymbol{\sigma}_h^T) \\
-d(\mathbf{u}_h, \boldsymbol{\sigma}_h) - d(\mathbf{u}_h, \boldsymbol{\sigma}_h^T) &= r_3(\boldsymbol{\sigma}_h).
\end{aligned}$$

For the trilinear form of the convective term in the momentum equations, we use a Lax-Friedrichs flux (Shahbazi et al., 2007):

$$\begin{aligned}
c(\mathbf{w}_h, \mathbf{u}_h, \mathbf{v}_h) &= - \int_{\Omega_h} \text{Re}(\mathbf{u}_h \otimes \mathbf{w}_h) : \nabla_h \mathbf{v}_h \, dV \\
&\quad - \oint_{\Gamma \setminus \Gamma_D} \text{Re} \left(\{\mathbf{u}_h \otimes \mathbf{w}_h\} \mathbf{n}_\Gamma - \frac{\gamma_1}{2} \llbracket \mathbf{u}_h \rrbracket \right) \cdot \llbracket \mathbf{v}_h \rrbracket \, dS. \quad (4.50)
\end{aligned}$$

The local Lax-Friedrich parameter is defined as in Shahbazi et al. (2007). It is chosen to be:

$$\gamma_1 := \max \left\{ |\lambda|; \lambda \in \text{spec} \left(\mathbf{M} \left(\overline{\mathbf{u}^-} \right) \right) \cup \text{spec} \left(\mathbf{M} \left(\overline{\mathbf{u}^+} \right) \right) \right\} \quad (4.51)$$

with $\text{spec}(M)$ is the spectrum of the tensor M , $\overline{\mathbf{u}^-}$ and $\overline{\mathbf{u}^+}$ the mean values \mathbf{u}^- and \mathbf{u}^+ respectively, of two adjacent cells sharing a common edge and

$$\mathbf{M}(\mathbf{u}) := \nabla_{\mathbf{u}} ((\mathbf{u} \otimes \mathbf{u}) \mathbf{n}_\Gamma). \quad (4.52)$$

In other words, the penalty is the largest eigenvalue of the Jacobian defined by Eq. (4.52) of the mean values of \mathbf{u} of two adjacent elements in an absolute sense.

For the bilinear form of the pressure gradient as well as for the velocity divergence in the continuity equation we use

$$b(p_h, \mathbf{v}_h) = - \int_{\Omega_h} p_h \nabla_h \cdot \mathbf{v}_h \, dV - \oint_{\Gamma \setminus \Gamma_N} \llbracket \mathbf{v}_h \rrbracket \cdot \mathbf{n}_\Gamma \{p_h\} \, dS. \quad (4.53)$$

The Laplacian in the momentum equations is discretized using the Symmetric Interior Penalty (SIP) method (Shahbazi, 2005):

$$\begin{aligned}
a(\mathbf{u}_h, \mathbf{v}_h) &= - \int_{\Omega_h} \beta (\nabla_h \mathbf{u}_h : \nabla_h \mathbf{v}_h) \, dV + \oint_{\Gamma \setminus \Gamma_N \setminus \Gamma_S} \beta \{ \nabla_h \mathbf{u}_h \} \mathbf{n}_\Gamma \cdot \llbracket \mathbf{v}_h \rrbracket \, dS \\
&\quad + \oint_{\Gamma \setminus \Gamma_N \setminus \Gamma_S} \beta \{ \nabla_h \mathbf{v}_h \} \mathbf{n}_\Gamma \cdot \llbracket \mathbf{u}_h \rrbracket \, dS - \oint_{\Gamma \setminus \Gamma_N \setminus \Gamma_S} \gamma_2 \llbracket \mathbf{u}_h \rrbracket \cdot \llbracket \mathbf{v}_h \rrbracket \, dS + a_S(\mathbf{u}_h, \mathbf{v}_h). \quad (4.54)
\end{aligned}$$

As stated in Arnold (1982), the penalty parameter γ_2 has to be chosen large enough to ensure coercivity while it should be as small as possible since over penalization increases the approximation error and enhances the condition number of the system. Ensuring coercivity means:

$$a(\mathbf{u}_h, \mathbf{u}_h) \geq \text{const} \|\mathbf{u}\|_*^2 \quad \forall \mathbf{u} \in \mathbb{P}_k(\mathcal{R}) \quad (4.55)$$

for any suitable norm $\| - \|_*$ (Kummer, 2017). In case of the SIP method, the coercivity is reached by bounding the norms on the element edge by norms within the element. The so-called inverse trace inequalities are defined as:

$$\|\mathbf{u}\|_{L^2(\partial K)}^2 \leq \frac{C(k)}{h} \|\mathbf{u}\|_{L^2(K)}^2 \quad \forall \mathbb{P}_k(\mathfrak{K}) \quad (4.56)$$

with $C(k) \sim \mathcal{O}(k^2)$ (Shahbazi, 2005). Thus,

$$\begin{aligned} \gamma_2 &:= \max \left\{ \beta \frac{\gamma_0 k^2}{h^-}, \beta \frac{\gamma_0 k^2}{h^+} \right\} \quad \text{on } \Gamma_I, \\ \gamma_2 &:= \beta \frac{\gamma_0 k^2}{h^-} \quad \text{on } \partial\Omega_h. \end{aligned} \quad (4.57)$$

The exact value of the geometric factor $\frac{1}{h}$ is dependent of the problem and the element type used. Note that for a Cartesian equidistant grid γ_2 is identical for all cells.

The free-slip BC for Γ_S in the SIP flux $a_S(\mathbf{u}_h, \mathbf{v}_h)$ reads:

$$\begin{aligned} a_S(\mathbf{u}, \mathbf{v}) &= \oint_{\Gamma_S} \beta \mathbf{n}_{\Gamma_S} \cdot (\{\nabla_h \mathbf{u}_h\} \mathbf{n}_{\Gamma_S}) \llbracket \mathbf{v}_h \rrbracket \cdot \mathbf{n}_{\Gamma_S} \, dS \\ &\quad + \oint_{\Gamma_S} \beta \mathbf{n}_{\Gamma_S} \cdot (\{\nabla_h \mathbf{v}_h\} \mathbf{n}_{\Gamma_S}) \llbracket \mathbf{u}_h \rrbracket \cdot \mathbf{n}_{\Gamma_S} \, dS \\ &\quad - \oint_{\Gamma_S} \gamma_2 \llbracket \mathbf{u}_h \rrbracket \cdot \mathbf{n}_{\Gamma_S} \llbracket \mathbf{v}_h \rrbracket \cdot \mathbf{n}_{\Gamma_S} \, dS. \end{aligned} \quad (4.58)$$

For the stress divergence in the momentum equations and the gradient of velocity in the constitutive equations we have structurally the same weak formulation, for the stress divergence it reads:

$$d'(\boldsymbol{\tau}_h, \mathbf{v}_h) = - \int_{\Omega_h} \boldsymbol{\tau}_h : \nabla_h \mathbf{v}_h \, dV + \oint_{\Gamma \setminus \Gamma_N \setminus \Gamma_S} \{\boldsymbol{\tau}_h\} \mathbf{n}_{\Gamma} \cdot \llbracket \mathbf{v}_h \rrbracket \, dS + d_S(\boldsymbol{\tau}_h, \mathbf{v}_h), \quad (4.59)$$

and for the gradient of velocity in the constitutive equations it is:

$$d(\mathbf{u}_h, \boldsymbol{\sigma}_h) = - \int_{\Omega_h} (1 - \beta) \mathbf{u}_h \cdot \nabla_h \cdot \boldsymbol{\sigma}_h \, dV + \oint_{\Gamma} (1 - \beta) (\{\mathbf{u}_h\} \otimes \mathbf{n}_{\Gamma}) : \llbracket \boldsymbol{\sigma}_h \rrbracket \, dS. \quad (4.60)$$

In case of the stress divergence we need to define a free-slip BC $d_S(\boldsymbol{\tau}_h, \mathbf{v}_h)$ at Γ_S :

$$d_S(\boldsymbol{\tau}_h, \mathbf{v}_h) = \oint_{\Gamma_S} \mathbf{n}_{\Gamma_S} \cdot (\{\boldsymbol{\tau}_h\} \mathbf{n}_{\Gamma_S}) \llbracket \mathbf{v}_h \rrbracket \cdot \mathbf{n}_{\Gamma_S} \, dS. \quad (4.61)$$

The trilinear form for the convective term in the constitutive equations including the streamline upwinding is as mentioned in Sec. 3.4.2 (Fortin and Fortin, 1989):

$$\begin{aligned} f(\mathbf{w}_h, \boldsymbol{\tau}_h, \boldsymbol{\sigma}_h) &= - \int_{\Omega_h} \text{Wi}(\mathbf{w}_h \cdot \nabla_h \boldsymbol{\tau}_h) : \boldsymbol{\sigma}_h \, dV \\ &\quad - \oint_{\Gamma_I} \text{Wi} \{\mathbf{w}_h\} \cdot \mathbf{n}_{\Gamma_I} \llbracket \boldsymbol{\tau}_h \rrbracket : (f_2 \boldsymbol{\sigma}_h^+ - f_1 \boldsymbol{\sigma}_h^-) \end{aligned} \quad (4.62)$$

with the following functions depending on the upwind parameter $0 \leq \alpha \leq 1$:

$$\begin{aligned} f_1(\alpha) &:= \begin{cases} \alpha, & \text{if } \{\mathbf{w}_h\} \cdot \mathbf{n}_{\Gamma_I} < 0, \\ 1 - \alpha, & \text{if } \{\mathbf{w}_h\} \cdot \mathbf{n}_{\Gamma_I} \geq 0, \end{cases} \\ f_2(\alpha) &:= \begin{cases} \alpha, & \text{if } \{\mathbf{w}_h\} \cdot \mathbf{n}_{\Gamma_I} < 0, \\ 1 - \alpha, & \text{if } \{\mathbf{w}_h\} \cdot \mathbf{n}_{\Gamma_I} \geq 0. \end{cases} \end{aligned} \quad (4.63)$$

The objective terms of the constitutive equations consist of the following trilinear form:

$$g(\mathbf{w}_h, \boldsymbol{\tau}_h, \boldsymbol{\sigma}_h) = - \int_{\Omega_h} \text{Wi} (\nabla_h \mathbf{w}_h \cdot \boldsymbol{\tau}_h) : \boldsymbol{\sigma}_h \, dV. \quad (4.64)$$

In the LDG method we need a penalty flux to ensure the stability of the system (Cockburn et al., 2002). The bilinear form reads:

$$s_1(\mathbf{u}_h, \mathbf{v}_h) = - \oint_{\Gamma \setminus \Gamma_N \setminus \Gamma_S} \frac{\gamma_3}{h_{\min}} \llbracket \mathbf{u}_h \rrbracket \cdot \llbracket \mathbf{v}_h \rrbracket \, dS. \quad (4.65)$$

with γ_3 is in our case chosen to be 1 in all simulations. The right-hand-sides of the problem include the Dirichlet boundary values and all values from the BDF2 discretization of time level t^n and t^{n-1} . The right-hand-side for the continuity equation is:

$$r_1(q_h) = - \oint_{\Gamma_D} q_h \mathbf{u}_D \cdot \mathbf{n}_{\Gamma_D} \, dS. \quad (4.66)$$

The right-hand-side of the momentum equations is:

$$\begin{aligned} r_2(\mathbf{v}_h) &= \oint_{\Gamma_D} ((\mathbf{u}_D \otimes \mathbf{u}_D) \mathbf{n}_{\Gamma_D \cup \Gamma_S} + \frac{\gamma_1}{2} \mathbf{u}_D) \cdot \llbracket \mathbf{v}_h \rrbracket \, dS \\ &\quad - \oint_{\Gamma_D} \mathbf{u}_D \cdot (\beta \{\nabla_h \mathbf{v}_h\} \mathbf{n}_{\Gamma_D} - \gamma_2 \llbracket \mathbf{v}_h \rrbracket) \, dS \\ &\quad - \oint_{\Gamma_D} \frac{\gamma_3}{h_{\min}} \mathbf{u}_D \cdot \llbracket \mathbf{v}_h \rrbracket \, dS + \frac{\text{Re}}{\Delta t} \int_{\Omega_h} \left(2\mathbf{u}^n - \frac{1}{2}\mathbf{u}^{n-1} \right) \cdot \mathbf{v}_h \, dV. \end{aligned} \quad (4.67)$$

It includes the Dirichlet BC for the convective part, for the Laplace term and for the penalty flux s_1 and the values for the velocities from the BDF2 discretization of time level t^n and t^{n-1} . The right-hand-side for the constitutive equations consists of a Dirichlet boundary for the velocities in the gradient of velocity and the values for the stresses from the BDF2 discretization of time level t^n and t^{n-1} :

$$\begin{aligned} r_3(\boldsymbol{\sigma}_h) &= - \oint_{\Gamma_D} (1 - \beta) \mathbf{u}_D \otimes \mathbf{n}_\Gamma : \llbracket \boldsymbol{\sigma}_h \rrbracket \, dS - \oint_{\Gamma_D} (1 - \beta) \mathbf{u}_D \otimes \mathbf{n}_\Gamma : \llbracket \boldsymbol{\sigma}_h^T \rrbracket \, dS \\ &\quad + \frac{\text{Wi}}{\Delta t} \int_{\Omega_h} \left(2\boldsymbol{\tau}^n - \frac{1}{2}\boldsymbol{\tau}^{n-1} \right) : \boldsymbol{\sigma}_h \, dV. \end{aligned} \quad (4.68)$$

Since the stress tensor is symmetric and we choose the test function to be $\boldsymbol{\sigma}_h = \boldsymbol{\sigma}_h^T$, we can omit the constitutive equation for τ_{21} and therefore, neglect all terms belonging to τ_{21} in the discretized system.

5. The EXTENDED Discontinuous Galerkin Method for Viscoelastic Droplets

In this chapter we present the eXTENDED DG discretization, also called unfitted or cut-cell DG, for multi-phase flows. In the BoSSS framework a sharp interface approach using a level-set function is implemented. The methods presented in this chapter are mainly deduced from the work of Kummer (2017) for the spatial discretization and Kummer et al. (2018) for the time integration of the level-set. Since the focus of this work was the discretization of the constitutive equations for the viscoelastic flow and defining the numerical fluxes at the interface for the viscoelastic multi-phase flow, the statements in this chapter are kept short. For a more detailed information about the treatment of the level-set and its evolution as well as the topological treatment of the cut-cells by the level-set, the reader is referred to the detailed work from the BoSSS working group on multi-phase flow discretization (e.g. Kummer, 2017; Utz et al., 2017; Kummer et al., 2018; Kummer et al., 2020b).

5.1. State of Knowledge

The XDG method was developed from the eXTENDED Finite Element Method (XFEM, e.g. Sauerland and Fries, 2011; Sauerland and Fries, 2013) thanks to the close proximity of both methods. Further, Lehrenfeld and Reusken (2012) developed several techniques for XDG method and eXTENDED Finite Element Method (XFEM) using a level-set function to describe the interface between two phases (e.g. Lehrenfeld and Reusken, 2012; Lehrenfeld and Reusken, 2013; Lehrenfeld, 2016).

Heimann et al. (2013) presented in their study a full XDG discretization for the two-phase Navier-Stokes equations using a piecewise linear approximation of the interface and an interior penalty method to describe the velocity and pressure jump on the interface with high accuracy. Kummer and Oberlack (2013) also presented a full DG solver for single-phase and two-phase flow using an XDG method with a level-set function. The full XDG spatial discretization and the scalability of the framework for a XDG Poisson problem are presented in Kummer (2017) and Kummer et al. (2020b).

5.2. Definitions for the Extended Discontinuous Galerkin Space

The definitions of this section are mainly derived from Kummer et al. (2020b). The definitions from sec. 4.2 are extended at this point for multi-phase flow and include the time dependency

of the interface $\mathcal{I}(t)$.

- The physical domain $\Omega \subset \mathbb{R}^2$ with a boundary $\partial\Omega$ is divided in the case of multi-phase flow into two disjoint but adjacent subdomains $\mathfrak{A}(t)$ and $\mathfrak{B}(t)$ which can be time-dependent.
- We introduce an interface $\mathcal{I}(t)$ which can be time-dependent and which we demand to be at least C^1 -continuous almost everywhere, resulting in the decomposition $\Omega := \mathfrak{A}(t) \cup \mathcal{I}(t) \cup \mathfrak{B}(t)$.
- We extend the numerical grid \mathfrak{K} by the time-dependent cut-cells $K_{j,\mathfrak{A}}(t) := K_j \cap \mathfrak{A}(t)$ and $K_{j,\mathfrak{B}}(t) := K_j \cap \mathfrak{B}(t)$, respectively. Thus, we define the set of all time-dependent cut-cells as $\mathfrak{K}^X(t)$. If a general cut-cell of either species or phase \mathfrak{A} or \mathfrak{B} is considered, we denote it with $K_{j,s}(t)$.
- The union of all edges is extended by the interface such that $\Gamma := \bigcup_j \partial K_j \cup \mathcal{I}$.
- We extend the normal field to $\mathbf{n}_{\mathcal{I},\Gamma}$ which is equal to $\mathbf{n}_{\mathcal{I}}$ on $\mathcal{I}(t)$ and equal to \mathbf{n}_{Γ} on Γ .
- The information in the interior of an element and the exterior information of the neighbouring cell is extended for a field $\psi \in C^0(\Omega_h \setminus \Gamma_I \setminus \mathcal{I})$ with:

$$\psi^- := \lim_{\varepsilon \searrow 0} \psi(\mathbf{x} - \varepsilon \mathbf{n}_{\mathcal{I},\Gamma}) \quad \text{for } \mathbf{x} \in \Gamma \cup \mathcal{I} \quad (5.1)$$

$$\psi^+ := \lim_{\varepsilon \searrow 0} \psi(\mathbf{x} + \varepsilon \mathbf{n}_{\mathcal{I},\Gamma}) \quad \text{for } \mathbf{x} \in \Gamma_I \cup \mathcal{I} \quad (5.2)$$

- We extend the broken polynomial space of total degree k (Eq. (4.8)) to the XDG space $\mathbb{P}_k^X(\mathfrak{K}(t)) := \mathbb{P}_k(\mathfrak{K}^X(t))$.
- We redefine the function space for test and trial functions using a time dependent broken polynomial space:

$$\mathbb{V}_{\mathbf{k}}^X(t) := \prod_{i=1}^{D_v} \mathbb{P}_{k_i}^X(\mathfrak{K}, t). \quad (5.3)$$

Therefore, the local and global inner product and the local and global $L^2(\mathfrak{K})$ norm are defined for $u_K, v_K \in \mathbb{V}_{\mathbf{k}}^X(t)$ and $u_h, v_h \in \mathbb{V}_{\mathbf{k}}^X(t)$, respectively.

- The broken gradient $\nabla_h \mathbf{u}_h$ and the broken divergence $\nabla_h \cdot \mathbf{u}_h$ on the domain Ω_h exclude differentiation at $\Gamma \cup \mathcal{I}(t)$.

For a two-phase conservation equation of the form:

$$\frac{\partial u}{\partial t} + \nabla \cdot f(u) = 0 \quad \text{for } u \in \Omega \setminus \mathcal{I}, \quad (5.4)$$

$$\llbracket f(u) \rrbracket \cdot \mathbf{n}_{\mathcal{I}} = 0 \quad \text{on } \mathcal{I}, \quad (5.5)$$

the corresponding minimization problem, analogously to Eq. (4.21), reads: find $u_h(\mathbf{x}, t) \in \mathbb{V}_{\mathbf{k}}^X(t)$ such that $\forall v_h(\mathbf{x}, t) \in \mathbb{V}_{\mathbf{k}}^X(t)$:

$$\int_{\Omega_h} \frac{\partial u_h}{\partial t} v_h \, dV - \int_{\Omega_h} f_h(u_h) \nabla_h v_h \, dV = - \int_{\Gamma} f^*(u_h^-, u_h^+, \mathbf{n}_{\mathcal{I},\Gamma}) \llbracket v_h \rrbracket \, dV. \quad (5.6)$$

5.3. Interface Representation and Evolution

The XDG representation in Sec. 5.2 is incomplete without specifying the interface motion in time. In order to capture the interface within the flow, a sufficiently smooth level-set field φ with the following properties is introduced:

$$\begin{aligned}\mathfrak{A}(t) &:= \{\mathbf{x} \in \Omega : \varphi(\mathbf{x}, t) < 0\}, \\ \mathfrak{I}(t) &:= \{\mathbf{x} \in \Omega : \varphi(\mathbf{x}, t) = 0\}, \\ \mathfrak{B}(t) &:= \{\mathbf{x} \in \Omega : \varphi(\mathbf{x}, t) > 0\}.\end{aligned}\tag{5.7}$$

A commonly used function for φ is the signed distance function to the initial interface, representing the minimal distance of a given point \mathbf{x} to the interface \mathfrak{I} by the value of φ . The sign of the value determines if the point \mathbf{x} is in phase \mathfrak{A} or phase \mathfrak{B} (Gross and Reusken, 2011). If the interface position is given, this signed distance formulation is the solution of the Eikonal equation in a suitable domain Ω around the interface:

$$\begin{aligned}|\nabla\varphi| - 1 &= 0 \quad \text{on } \Omega, \\ \varphi &= 0 \quad \text{on } \mathfrak{I}^*\end{aligned}\tag{5.8}$$

for a predefined interface \mathfrak{I}^* . The movement of the interface is given by the evolutionary equation of the level-set field on the interface $\mathfrak{I}(t)$:

$$\frac{\partial\varphi}{\partial t} + \mathbf{u} \cdot \nabla\varphi = 0.\tag{5.9}$$

We consider the transport equation of the level-set (5.9) to be discretized in space and time (Sec. 5.4) by using $\mathbf{u} \cdot \nabla\varphi = \nabla \cdot (\mathbf{u}\varphi)$, exploiting the divergence free property of the velocity field.

To obtain a well-posed problem, we need a signed distance IC $\varphi_0(\mathbf{x}) := \varphi(\mathbf{x}, 0)$ at time $t_0 = 0$:

$$\begin{aligned}\mathfrak{A}(0) &= \{\mathbf{x} \in \Omega : \varphi_0(\mathbf{x}) < 0\}, \\ \mathfrak{I}(0) &= \{\mathbf{x} \in \Omega : \varphi_0(\mathbf{x}) = 0\}, \\ \mathfrak{B}(0) &= \{\mathbf{x} \in \Omega : \varphi_0(\mathbf{x}) > 0\}.\end{aligned}\tag{5.10}$$

Using the level-set, we can define the interface normal $\mathbf{n}_{\mathfrak{I}}$ as follows:

$$\mathbf{n}_{\mathfrak{I}} := \frac{\nabla\varphi}{|\nabla\varphi|}.\tag{5.11}$$

and rewrite Eq. (5.9):

$$\frac{\partial\varphi}{\partial t} + \mathbf{u} \cdot \mathbf{n}_{\mathfrak{I}} |\nabla\varphi| = 0\tag{5.12}$$

We still need to define the velocity of the interface in normal direction which should be equal to the flow velocity in normal direction since we have a material interface. The interface may only be moved with the velocity directly on the interface, although the level-set function is defined on the whole domain, we need to extend this velocity properly onto the domain while keeping the signed-distance property. Hence, we need to construct a velocity field \mathbf{u}^* which keeps the signed distance property and propagates the level-set in the domain Ω . This

extension problem will be discussed in detail in Sec. 5.3.2. However, on the one hand, the level-set can lose its signed distance property and needs to be re-initialised from time to time (Sec. 5.3.1). On the other hand, we cannot always guarantee the divergence free property and sufficient smoothness of the velocity field such that the extensional velocity is always exact and the level-set is propagated correctly and without discontinuities at the edges.

5.3.1. Elliptic Re-initialisation

Depending on the underlying velocity field the level-set can become very steep or flat close to the interface during evolution, so the goal is to restore the signed distance properties of the level-set which means finding a solution for Eq. (5.8) with the actual interface position. This means the re-initialisation is an algorithm to reduce errors in the position of the zero-level-set due to the discretization of the level-set. During time evolution, the deviation of the discrete level set in the neighbourhood of the interface from a signed distance function is traced. As an indicator we use the L^2 norm of the gradient of the discrete level-set. If the deviation becomes greater than a defined tolerance value the discrete level-set must be re-initialized (Gross and Reusken, 2011). Starting point for the re-initialisation is an initial function φ^* that defines the interface $\mathcal{I}^* = \mathcal{I}(\varphi)$ and does not have signed distance property (Utz et al., 2017).

BoSSS utilizes a global PDE based approach to reformulate and solve the problem (5.8). The re-initialisation rewrites the hyperbolic problem, emerging by directly introducing the signed distance property and Eikonal equation into the level-set evolution, into an elliptic problem. This avoids oscillations in the zero iso-contour and stability issues.

In detail, we consider the energy functional $\mathcal{E}(\varphi)$ using a single-well function as a potential:

$$\mathcal{E}(\varphi) = \frac{1}{2} \int_{\Omega} (|\nabla\varphi| - 1)^2 \, d\mathbf{x}. \quad (5.13)$$

We then have to solve the minimization problem

$$\min \mathcal{E}(\varphi), \quad \text{subject to } \varphi = 0 \text{ on } \mathcal{I}(\varphi^*) \quad (5.14)$$

where the constraint at the interface can be enforced by a penalty term

$$P(\varphi^*, \varphi) = a \int_{\mathcal{I}(\varphi^*)} \frac{\varphi^2}{2} \, dS. \quad (5.15)$$

The discussion about the choice of the value for a , which is constructed identically as the penalty in the SIP method, can be found in Utz et al. (2017). Hence, we can find a weak formulation for the minimization problem using the Gâteaux derivative of both terms:

$$\delta\mathcal{E}(\varphi, v) + \delta P(\varphi, v) = \int_{\Omega} d(|\nabla\varphi|) \nabla\varphi \cdot \nabla v \, d\mathbf{x} + \alpha \int_{\mathcal{I}(\varphi^*)} \varphi v \, dS = 0 \quad \forall v \quad (5.16)$$

with v being the test function.

5.3.2. The Extensional Problem

We reconstruct the underlying velocity \mathbf{u} of the fluid field by the extensional velocity u^* for the level-set evolution in two steps. First, its value and direction is evaluated in the cells directly cut by the interface (the cut-cells, more details can be found in Sec. 5.5). Afterwards, the velocity is propagated to the direct neighbours of the cut-cells which is called the narrow band. Since we have a capillary time restriction like the CFL for the size of a time step ensuring that the level-set does not move further than from one to another cell, the information about the velocity in the narrow band is sufficient. Outside of the region the level-set is set to 1 or -1 , respectively, depending on the fluid phase \mathfrak{A} or \mathfrak{B} . Subsequently, the level-set can be moved using this velocity field. In BoSSS the extension occurs twice in the two-dimensional case, each with the velocity u_x and the velocity u_y as extensional velocity u^* . We choose the extensional velocity field u^* at point \mathbf{x} to be the value at the closest point \mathbf{x}_c on the interface:

$$u^*(\mathbf{x}) := u_0(\mathbf{x}_c). \quad (5.17)$$

This is the solution of a PDE

$$\begin{aligned} \nabla\varphi \cdot (\nabla u^*) &= 0 && \text{in } \Omega, \\ u^* &= u_0 && \text{on } \mathfrak{I}. \end{aligned} \quad (5.18)$$

Extension on Cut-Cells

For the extension of the velocity in the cut-cells we use a similar elliptic global based PDE approach like in the elliptic re-initialisation of the level-set. We can again reformulate Eq. (5.18) as a minimization problem of a potential function:

$$\begin{aligned} \min \mathcal{E}(u) &= \int_{\Omega} \text{pot}(\nabla\varphi \cdot \nabla u) \, dV, \\ \text{subject to } u &= u_0 \quad \text{on } \mathfrak{I} \end{aligned} \quad (5.19)$$

with its minimum at $\text{pot}(0)$. The simplest choice is $\text{pot}(y) = \frac{y^2}{2}$ (Utz and Kummer, 2018). Reformulating this problem in a variational formulation, using again the Gâteaux derivative and a Lagrange multiplier ξ and performing integration by parts leads to the weak formulation of the problem:

$$\begin{aligned} & - \int_{\Omega} \nabla \cdot ((\nabla\varphi \otimes \nabla\varphi)\nabla u)v \, dV \\ & + \int_{\partial\Omega} ((\nabla\varphi \otimes \nabla\varphi)\nabla u) \cdot \mathbf{n}v \, dV - \xi \oint_{\mathfrak{I}} (u - u_0)v \, dS = 0 \quad \forall v \end{aligned} \quad (5.20)$$

A detailed derivation of this equation can be found in Utz and Kummer (2018). Locally, we can find a PDE formulation of the problem:

$$\begin{aligned} \nabla \cdot ((\nabla\varphi \otimes \nabla\varphi)\nabla u) &= 0 && \text{in } \Omega \setminus \mathfrak{I}, \\ ((\nabla\varphi \otimes \nabla\varphi)\nabla u) \cdot \mathbf{n} &= 0 && \text{on } \partial\Omega, \\ u &= u_0 && \text{on } \mathfrak{I}. \end{aligned} \quad (5.21)$$

This is the steady state limit of an anisotropic diffusion of the velocity field. It has a Neumann BC and the Dirichlet BC $u = u_0$. The formulation we found is linear in u and therefore, can be solved after discretization using a direct solver (Utz and Kummer, 2018).

Extension in the Narrow Band

In the neighbouring elements of the cut-cells we have two options to find the extensional velocity. First, we can use the same PDE-based approach as in the cut-cells and second, we can use a fast-marching algorithm propagating the velocity found in the cut-cells into the narrow band using Eq. (5.17).

In this work we used the fast marching option which should be briefly described. Further information about the mathematical description can be found in Sussman and Hussaini (2003). Always favouring the locally optimal choice, fast marching algorithms iteratively extend the field point by point starting from the velocity in the cut-cells. The search algorithm starts outside of the cut-cells and finds the cell with the smallest mean value of the level-set. This cell is the closest cell to the zero level-set as we have ensured the signed distance property of the level-set function. Knowing the extensional velocity values in the quadrature nodes at least at one edge of the cell, the extension velocity inside the cell is calculated and set at the quadrature nodes of the cell. Hereby, the value from the edge is chosen with the closest distance to the quadrature node (closest point). After setting the values in the cell at all quadrature nodes, the status of this cell is set to 'accepted' which means that in this cell the values may not be changed again. This is repeated until the extensional velocity is set in all cells of the narrow band, and thus, all cells in the narrow band have the status 'accepted'.

5.4. Time Integration with a Moving Interface

For the discussion of the time integration in an XDG method with a moving interface we consider a conservation equation for a scalar quantity ψ including an interface moving with a velocity u in the normal direction of the interface $\mathbf{n}_{\mathcal{I}}$, which we describe using the Rankine-Hugoniot-condition (Kummer et al., 2018):

$$\begin{aligned} \frac{\partial \psi}{\partial t} + \nabla \cdot \mathbf{f}(\psi) &= 0 \quad \text{in } \Omega \setminus \mathcal{I}, \\ -u[[\psi]] + [[\mathbf{f}]] \cdot \mathbf{n}_{\mathcal{I}} &= 0 \quad \text{on } \mathcal{I}. \end{aligned} \tag{5.22}$$

We consider the temporal discretization from time level t_0 to t_1 . There are two options of treating the interface motion. The first one is a splitting method in which the position of the interface is kept constant during one time step and moved afterwards which requires an extrapolation from the old cut-cell mesh to the new one. It is easy to implement, but not conservative, inherently limited to low orders of accuracy, and extremely small time steps are needed. The second one is a moving interface approach using a space-time discretization such that the interface motion is taken into account during the time step. This method is conservative, allow for large time steps, and can be generalized to an arbitrary order of accuracy, but the system to be solved is severely larger and thus, is more expensive in computational memory (Kummer et al., 2018).

In both strategies topology changes in the cut-cell mesh can happen when the interface enters or leaves a cell. In case of the splitting method this change can be handled by using a polynomial interpolation. For the moving interface approach a cell-agglomeration algorithm presented in Sec. 5.5.1 is used. Thus, without loss of generality, we assume at this point that

there is no topological change between the cut-cell meshes $\mathfrak{K}^X(t_0)$ and $\mathfrak{K}^X(t_1)$. This means that if a cell has a zero or non-zero volume $|K_{j,s}(t_0)|$ at t_0 it must be zero or non-zero at time level t_1 , respectively. Hence, we investigate the time integration for a single cell $K_{j,s}(t)$ (Kummer et al., 2018).

In the moving interface approach we need a space-time description of the problem as illustrated in Fig. 5.1 and need to redefine some notations:

- A space-time element: $K_{j,s}^* := \{(t, \mathbf{x}) \mid t_0 < t < t_1, \mathbf{x} \in K_{j,s}(t)\}$.
- A space-time boundary:

$$\partial K_{j,s}^* := (\{t_0 \times K_{j,s}(t_0)\}) \cup (\{t_1 \times K_{j,s}(t_1)\}) \cup (\{(t, \mathbf{x}) \mid t_0 \leq t \leq t_1, \mathbf{x} \in \partial K_{j,s}(t)\})$$

- then the space-time divergence is: $\nabla^* \cdot (\psi, \mathbf{f}(\psi)) := \frac{\partial \psi}{\partial t} + \nabla \cdot \mathbf{f}(\psi)$.
- The space-time outer normal is:

$$\mathbf{n}^* := \begin{cases} \frac{1}{\sqrt{1+u^{*2}}}(-u^*, \mathbf{n}_{\mathcal{I},\Gamma})^T & \text{on } \mathfrak{s} \cap \partial K_{j,s}^*, \\ (-1, 0)^T & \text{on } \{t_0\} \times \partial K_{j,s}, \\ (1, 0)^T & \text{on } \{t_1\} \times \partial K_{j,s}, \\ (0, \mathbf{n}_{\mathcal{I},\Gamma})^T & \text{elsewhere.} \end{cases}$$

The following integral identities for an integrand $f(t, \mathbf{x})$ are defined to use space-time notation in a form making it possible to use standard time integration schemes such as the BDF schemes described in Sec. 4.5. The volume integral is defined as:

$$\int_{K_{j,s}^*} f \, dV^* := \int_{t_0}^{t_1} \int_{K_{j,s}(t)} f \, dV \, dt \quad (5.23)$$

and on a boundary it is:

$$\oint_{\partial K_{j,s}^*} f \, dS^* := \int_{t_0}^{t_1} \oint_{\partial K_{j,s}(t)} f \sqrt{1+u^{*2}} \, dS \, dt + \int_{K_{j,s}(t_0)} f|_{t_0} \, dV - \int_{K_{j,s}(t_1)} f|_{t_1} \, dV. \quad (5.24)$$

With these definitions one can use the classical DG ansatz for discretization of different terms in space-time cut-cell $K_{j,s}^*$. At this point we present no further details and refer to the work of Kummer et al. (2018) presenting the methods in the context of the BoSSS implementation.

5.5. Spatial Discretization in Cut-Cells

If we have an interface moving through the computational domain Ω_h represented by a level-set function φ , we have a continuously changing set of cells which are separated into two parts by the zero level-set which we refer to as cut-cells. In these cut-cells we have the following local polynomial representation of a field, e.g. u :

$$u_K(\mathbf{x}, t) := \sum_{i=0}^k \hat{u}_{\mathcal{Q},i}(t) \phi_i(\mathbf{x}) \mathbf{1}_{\mathcal{Q}}(\varphi) + \hat{u}_{\mathcal{B},i}(t) \phi_i(\mathbf{x}) \mathbf{1}_{\mathcal{B}}(\varphi) \quad (5.25)$$

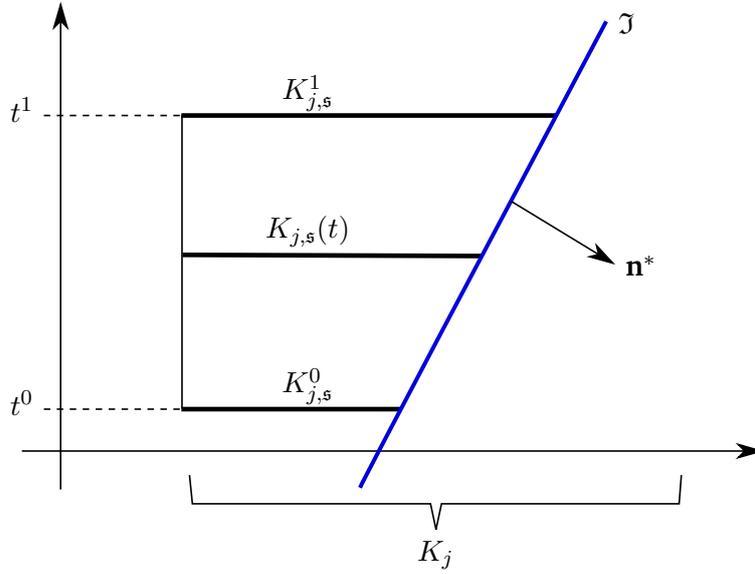


Figure 5.1.: Sketch of a space-time cut-cell $K_{j,s}^*$ for a constant interface velocity u^* (Kummer et al., 2018).

with $1_{\mathfrak{A}} := H(-\varphi(\mathbf{x}, t))$, $1_{\mathfrak{B}} := H(\varphi(\mathbf{x}, t))$ the Heaviside functions for fluid phase \mathfrak{A} and \mathfrak{B} , respectively.

These cut-cells need special handling by defining the integration over these cells. There are three major issues: First, appearing jumps in the interface due to the discontinuity between neighbouring cut-cells must be handled. Second, small cut-cells can undesirably increase the condition number of the problem and topology changes between time steps must be avoided for time integration. Last, an appropriate quadrature rule for the integration on cut-cells must be found. The latter two issues are treated in the following sections. For the first, a constrained continuity projection of the level-set is used. This means that a projection in the L_2 sense onto a continuous DG field is used and continuity constraints, e.g. equality at the cell edge, are introduced. Different approaches for such constraints are currently under development in the BoSSS working group. One example for such constraints is the path recovery filtering, applied on the curvature computation, presented in Kummer and Warburton (2016).

5.5.1. Cell Agglomeration

In case of multi-phase flow the interface moves through a stationary grid, producing a time-dependent cut-cell mesh $\mathfrak{K}^X(t)$. Since the interface position within the mesh is arbitrary, the cut-cells can become arbitrary small. These small cut-cells lead e.g. to large penalty parameters like in the SIP terms and hence, can cause undesirably high condition numbers (Kummer et al., 2020b). A good measure for the size of the cut-cell is the volume fraction with respect to the background cell $\frac{|K_{j,s}(t)|}{K_j}$. To minimize these effects the cell agglomeration algorithm implemented in BoSSS prevents the formation of these small cut-cells. Furthermore, the algorithm ensures that there are no topology changes in the cut-cell mesh in one time step, since the cut-cell mesh at time t_0 and t_1 need to have equal topology in order to have a

well-defined method. Otherwise the lifting operator in the space-time environment, defined in Sec. 5.4, is undefined. Obviously for multi-step time discretization schemes the topology has to be the same for all time levels (Kummer et al., 2020b).

Bringing all together the cell agglomeration algorithm must fulfil three requirements:

- The agglomerated meshes of t_0 and t_1 have the same topology.
- All cut-cells with a volume fraction of $0 < \frac{|K_{j,s}(t)|}{|K_j|} \leq a$ with a predefined a are agglomerated.
- There is no agglomeration across species.

In the implementation there is a loop over all cut-cells and the cut-cell is agglomerated to the largest neighbor in the same species if it is a new cell ($|K_{j,s}(t_1)| > 0$ and $|K_{j,s}(t_0)| = 0$) or a vanished cell ($|K_{j,s}(t_1)| = 0$ and $|K_{j,s}(t_0)| > 0$) or its volume fraction defined above is below the threshold $a \approx 0.1 \dots 0.2$ (Fig. 5.2).

We consider a set of all logical edges $Edge(\mathfrak{R}^X)$. A logical edge is a common edge of two adjacent cells K_j and K_i for which applies $\oint_{\overline{K_j \cap K_i}} 1 \, dS > 0$. With the agglomerated cut-cell mesh we need to redefine the polynomial space:

$$\mathbb{P}_k^{X,agg}(\mathfrak{R}^X(t)) := \mathbb{P}_k^X(Agg(\mathfrak{R}^X(t), A_M)) \quad (5.26)$$

with $A_M \subset Edge(\mathfrak{R}^X(t_1))$ being the aggregation map as a subset from all logical edges $Edge(\mathfrak{R}^X)$ and $Agg(\mathfrak{R}^X(t), A_M)$ being the set of all aggregated cells which can be formed with A_M . Obviously, this agglomerated XDG space is a sub-space of the original XDG space. Consequently, instead of solving the variational system on the space $\mathbb{V}_k^X(t_1)$ it will now be solved on the aggregate space at time t_1 :

$$\mathbb{V}_k^X(t_1) := \prod_{i=1}^{D_v} \mathbb{P}_k^{X,agg}(\mathfrak{R}^X(t_1)) \quad (5.27)$$

While solving, the dependence on time t is dropped since all temporally evolving parts are solved for the new time level t_1 . After completion of the time step, the solution is transmitted to the non-agglomerated space and for the next time step it is projected back onto the potentially different agglomerated space to serve as an initial value. All details about this method, especially the mathematical theory and the graph colouring for the aggregation map, can be found in Kummer (2017).

5.5.2. Implicit High-Order Quadrature for Curved Surfaces

At this point we present a high-order accurate numerical quadrature method for the numerical integration of integrals over implicitly defined curved surfaces and volumes which is used in the BoSSS algorithm for the integration in cut-cells by the interface and was presented by Saye (2015). The implementation in BoSSS was conducted during a master thesis (Beck, 2018). The remarks on the method in this section are kept short, for a more detailed revision the reader is referred to the two publications mentioned above.

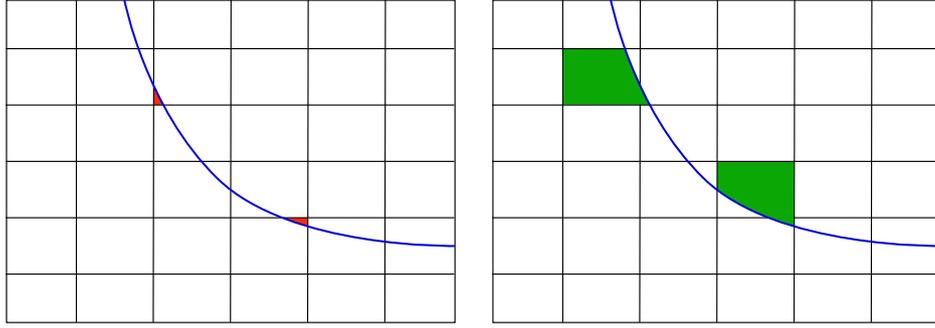


Figure 5.2.: Cell Agglomeration in a two-dimensional equidistant Cartesian grid. The blue curve is the interface with $\varphi = 0$ cutting the cells in which it is located. If a cut-cell is smaller than a threshold a (red cells, left) it is agglomerated to its largest neighbour (green cells, right).

First, we introduce a quadrature scheme for the volume integral and the surface integral of the form:

$$\int_{K_{j,s}} f \, dV \approx \sum_i \hat{w}_i f(x_i) \quad \text{and} \quad \int_{\mathcal{J} \cap K_{j,s}} g \, dS \approx \sum_k \tilde{w}_k g(\tilde{x}_k) \quad (5.28)$$

with some weights $\hat{w}_i, \tilde{w}_k \in \mathbb{R}$ and quadrature nodes $x_i, \tilde{x}_k \in \mathbb{R}^d$ where d is the dimension of the problem, in our case always two. Such a quadrature scheme leads to a convergence rate of order $\mathcal{O}(q_N^d)$ for the volume integrals and $\mathcal{O}(q_N^{d-1})$ for a surface integral with q_N being the amount of quadrature nodes. The method presented creates an algorithm by recursively rewriting the integrals until they can be easily numerically integrated by standard methods such as Gaussian quadrature. This is reached by converting the implicitly defined interface \mathcal{J} using a level-set function φ into a graph of an implicitly defined height function $\tilde{h}(\tilde{x})$ which we define later. Here, the theorem of multi-variable calculus guarantees the existence of such a height function which represents the interface in the cell $\mathcal{J} \cap K_j$ as a graph of $\tilde{h}(\tilde{x})$ (Fig. 5.3). Therefore, we need some definitions.

- We assume the cell $K_j = (x_1^L, x_1^U) \times \dots \times (x_d^L, x_d^U) \subset \mathbb{R}^d$ to be a hyper-rectangle
- and with the restriction function $\text{res}_i : \mathbb{R}^d \rightarrow \mathbb{R}^{d-1}$
with $(x_1, \dots, x_i, \dots, x_d) \mapsto (x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_d)$
we define the restricted cell $\tilde{K}_j := \text{res}_i(K_j)$.
- We have a point $\tilde{x} \in \tilde{K}_j : \tilde{x} = (x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_d)$.
- The search direction is e_i such that $x = \tilde{x} + ye_i$
- and the height function is $\tilde{h}(\tilde{x})$ which represents $\mathcal{J} \cap K_j$ as a graph of $\tilde{h}(\tilde{x})$, such that $\varphi(x_1, \dots, x_{i-1}, \tilde{h}(\tilde{x}), x_{i+1}, \dots, x_d) = 0$.
- We have then a restriction to the upper and lower parts of the function Φ_i^L, Φ_i^U with $\Phi_i^L := \varphi(\tilde{x} + x_i^L e_i)$ and $\Phi_i^U := \varphi(\tilde{x} + x_i^U e_i)$
- and a set $I_i \in \mathbb{R}$ with

$$I_i := \begin{cases} \{y \in (x_i^L, x_i^U) : s_i \varphi(\tilde{x} + ye_i) > 0\}, & \text{if } s_i = \pm 1, \\ \{y \in (x_i^L, x_i^U) : \varphi(\tilde{x} + ye_i) \neq 0\}, & \text{if } s_i = 0, \end{cases} \quad (5.29)$$

where the sign indicator of the evaluated domain is

$$s_i := \begin{cases} 1 & \text{if } \varphi < 0 \quad (\mathfrak{A}), \\ 0 & \text{if } \varphi = 0 \quad (\mathfrak{J}), \\ -1 & \text{if } \varphi > 0 \quad (\mathfrak{B}). \end{cases} \quad (5.30)$$

- We need a disjoint partitioning $\hat{V} = V_L \dot{\cup} V_U \subseteq \tilde{K}_j$ where

$$\begin{aligned} V_L &:= \{\Phi_i^L(\tilde{x}) : s_i \Phi_i^L(\tilde{x}) < 0\}, \\ V_U &:= \{\Phi_i^U(\tilde{x}) : s_i \Phi_i^U(\tilde{x}) < 0\} \cup \{\Phi_i^U(\tilde{x}) : s_i \Phi_i^U(\tilde{x}) > 0\}. \end{aligned} \quad (5.31)$$

We can then define the following volume and surface integrals by using a feasible height direction e_i and a disjoint partitioning \hat{V} :

$$\begin{aligned} \int_{K_{j,s}} f \, dV &= \int_{\hat{V}} \int_{I_i(\tilde{x})} f(\tilde{x} + ye_i) \, dy \, d\tilde{x}, \\ \int_{\mathfrak{J} \cap K_{j,s}} g \, dS &= \int_{\hat{V}} g \frac{|\nabla \varphi|}{\frac{\partial \varphi}{\partial x_i}} \Big|_{I_i(\tilde{x})=\tilde{x}+\tilde{h}(\tilde{x})e_i} \, d\tilde{x}. \end{aligned} \quad (5.32)$$

Then, these integrals are well defined and the contained one-dimensional integral $\int_{I_i(x)}(\cdot) \, dy$ is integrable by Gaussian quadrature. The surface integral includes the curvature of the surface by the following substitution:

$$dS = \sqrt{1 + |\nabla_{\tilde{x}} \tilde{h}|^2} \, d\tilde{x} = \frac{|\nabla \varphi|}{\frac{\partial \varphi}{\partial x_i}} \, d\tilde{x} \quad (5.33)$$

Note, that the sets $I_i(\tilde{x})$ and \hat{V} allow conveniently to switch between the integration on positive and negative domains, which is important for the implementation of the recursive algorithm for Eq. (5.32). This algorithm is shown for a two-dimensional cell in Fig. 5.4. It searches in one direction dependent on the distributional behaviour for the interface (red arrow, Fig. 5.4) until it reaches a root of the level-set function and q_N quadrature points (a,b) on that edge (blue dots, Fig. 5.4). From these points the algorithm searches in the direction of the height function until it again reaches points with roots of the level-set function and distributes quadrature points (1,...,4) along these arrows in the area of the cell. The weights in the quadrature are then a product for the weights from the quadrature points in the first and second search direction, e.g. $\hat{w}_1 = w_a \cdot w_1$ or $\hat{w}_4 = w_b \cdot w_4$.

5.5.3. Adaptive Mesh Refinement at the Interface

Since an adaptive mesh refinement algorithm dividing the cells of a Cartesian grid following some defined refinement parameters is present in the BoSSS code it can also be used in case of an XDG discretization with cut-cells and a level-set. Here, we choose a refinement strategy based on the curvature κ of the interface and defined refinement levels. The refinement of the elements occurs in the two-dimensional case with a 1:4 division, so one element is divided into four equally sized elements. Coarsening follows the same strategy. The neighbouring elements are always adapted such that no transition from one element edge to the edges of four elements occur. The output of an initial refinement around a droplet in a two-dimensional setting using a refinement level of three is presented in Fig. 5.5.

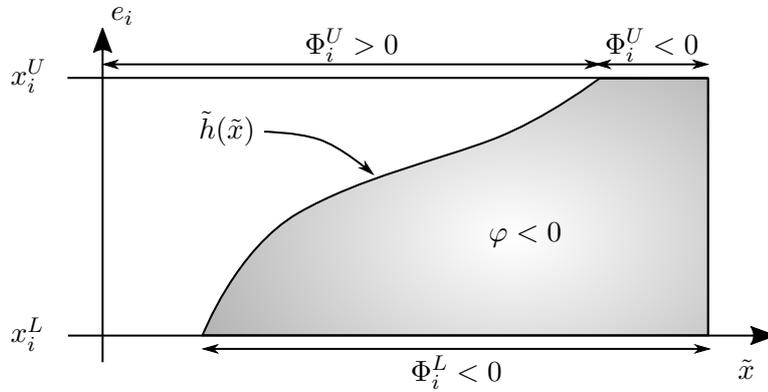


Figure 5.3.: Definition of Saye (2015) for the height function $\tilde{h}(\tilde{x})$ describing the zero level-set, here for the case that $\frac{\partial \varphi}{\partial x_i} < 0$ means it is beneath the graph of the height function, \tilde{x} abstractly represents a $d - 1$ -dimensional space.

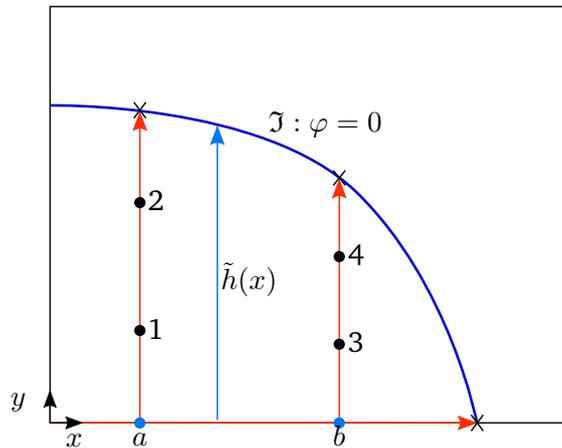


Figure 5.4.: Recursive strategy for the search of the quadrature points for a Gaussian quadrature for a single two-dimensional cell. The red arrows follow a search direction until a root of the level-set is reached and the quadrature nodes are distributed along their vector depending on the order of quadrature.

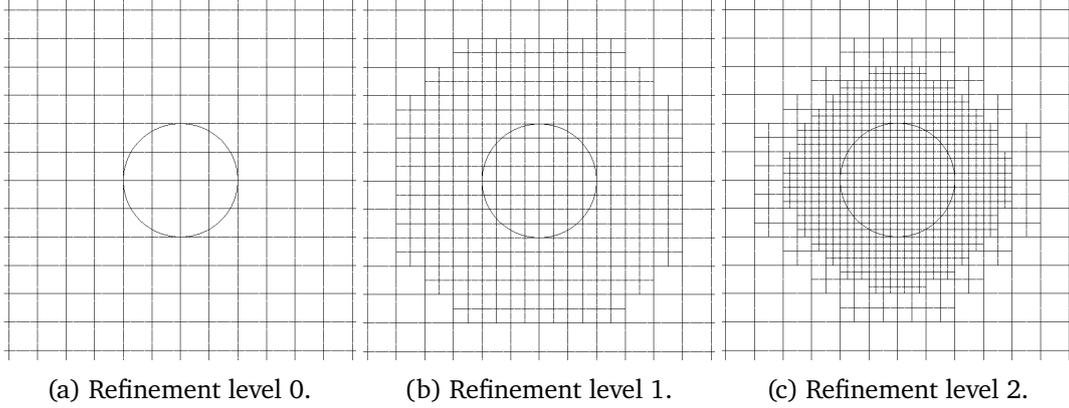


Figure 5.5.: Curvature refined adaptive mesh refinement along an interface for a droplet. The amount of refinement levels is 2.

5.6. Numerical Fluxes for the Two-Phase Setting

The discretization of the system of equations (2.66)-(2.68) presented in Sec. 4.6 can be easily extended for the multi-phase case with the specifications of Sec. 2.6 using the redefinitions from Sec. 5.2, particularly by using the redefined Γ . This means that now the fluxes at the interface are included in that discretization with the redefinition of the internal and external information at the interface. For the XDG discretization we use a slightly changed SIP method, adding the transposed term $(\nabla \mathbf{u})^T$ to the momentum equation:

$$\begin{aligned}
a(\mathbf{u}_h, \mathbf{v}_h)^{\text{XDG}} = & - \int_{\Omega_h} \beta (\nabla_h \mathbf{u}_h : \nabla_h \mathbf{v}_h + (\nabla_h \mathbf{u}_h)^T : \nabla_h \mathbf{v}_h) \, dV \\
& + \oint_{\Gamma \setminus \Gamma_N \setminus \Gamma_S} \beta \{ \nabla_h \mathbf{u}_h + (\nabla_h \mathbf{u}_h)^T \} \mathbf{n}_\Gamma \cdot \llbracket \mathbf{v}_h \rrbracket \, dS \\
& + \oint_{\Gamma \setminus \Gamma_N \setminus \Gamma_S} \beta \{ \nabla_h \mathbf{v}_h + (\nabla_h \mathbf{v}_h)^T \} \mathbf{n}_\Gamma \cdot \llbracket \mathbf{u}_h \rrbracket \, dS \\
& - \oint_{\Gamma \setminus \Gamma_N \setminus \Gamma_S} \gamma_2 \llbracket \mathbf{u}_h \rrbracket \cdot \llbracket \mathbf{v}_h \rrbracket \, dS + a_S(\mathbf{u}_h, \mathbf{v}_h)_S.
\end{aligned} \tag{5.34}$$

and consequently, for the free-slip BC for Γ_S in the SIP flux $a_S(\mathbf{u}_h, \mathbf{v}_h)$ reads:

$$\begin{aligned}
a_S(\mathbf{u}, \mathbf{v}) = & \oint_{\Gamma_S} \beta \mathbf{n}_{\Gamma_S} \cdot \left(\{ \nabla_h \mathbf{u}_h + (\nabla_h \mathbf{u}_h)^T \} \mathbf{n}_{\Gamma_S} \right) \llbracket \mathbf{v}_h \rrbracket \cdot \mathbf{n}_{\Gamma_S} \, dS \\
& + \oint_{\Gamma_S} \beta \mathbf{n}_{\Gamma_S} \cdot \left(\{ \nabla_h \mathbf{v}_h + (\nabla_h \mathbf{v}_h)^T \} \mathbf{n}_{\Gamma_S} \right) \llbracket \mathbf{u}_h \rrbracket \cdot \mathbf{n}_{\Gamma_S} \, dS \\
& - \oint_{\Gamma_S} \gamma_2 \llbracket \mathbf{u}_h \rrbracket \cdot \mathbf{n}_{\Gamma_S} \llbracket \mathbf{v}_h \rrbracket \cdot \mathbf{n}_{\Gamma_S} \, dS.
\end{aligned} \tag{5.35}$$

In the case of XDG the geometrical penalty parameter γ_2 also depends on cut-cells $K_{j,s}$, but there is no inverse trace inequality known for cut-cells of general shape. However, following the assumptions that in two dimensions the interface \mathcal{I} is approximately straight in one cut-cell $K_{j,s}$, which means that the local radius of the interface is large in the specific cell in comparison

to the size of the cell, and no cut-cell is cut twice by the interface, meaning $\mathcal{J} \cup K_j$ is simply connected we can find a dependency of all inequalities for different element shapes on the geometric factor $\frac{|\partial K_{j,s}|}{|K_{j,s}|}$. Both values of the nominator and denominator are known from the construction of the cut-cell quadrature rules in Sec. 5.5.2 and do not additionally need to be evaluated. So for the XDG case the local penalty factor is set to:

$$\tilde{\gamma}_2 := \gamma_0 k^2 \frac{|\partial K_{j,s}|}{|K_{j,s}|} \quad K_{j,s} \in \mathfrak{K}^X(t). \quad (5.36)$$

With the penalty factor being piecewise constant on each cut-cell it is defined as:

$$\begin{aligned} \gamma_2 &:= \max\{\beta^-, \beta^+\} \max\{\tilde{\gamma}_2^-, \tilde{\gamma}_2^+\} \quad \text{on } \Gamma_I, \\ \gamma_2 &:= \beta^- \tilde{\gamma}_2^- \quad \text{on } \partial\Omega_h. \end{aligned} \quad (5.37)$$

As has been shown by Kummer (2017) in numerical simulations, this choice for the penalty parameter for cut-cells seems to be suitable to ensure coercivity for the bilinear form of the SIP. He also showed that the condition number of the system is independent of the position of the interface \mathcal{J} (Kummer, 2017).

The flux for the stresses of the LDG method at the interface is treated as an inner edge flux using the dependent variables and material parameters from both phases. In the constitutive equation we must keep a free boundary since the material properties may only depend on the jumps in the material parameters. Therefore, we have implemented the following bilinear form including the flux for the gradient of velocity:

$$\begin{aligned} d(\mathbf{u}_h, \boldsymbol{\sigma}_h) &= - \int_{\Omega_h} (1 - \beta) \mathbf{u}_h \cdot \nabla_h \cdot \boldsymbol{\sigma}_h \, dV + \oint_{\Gamma \setminus \mathcal{J}} (1 - \beta) (\{\mathbf{u}_h\} \otimes \mathbf{n}_\Gamma) : \llbracket \boldsymbol{\sigma}_h \rrbracket \, dS \\ &\quad + \oint_{\mathcal{J}} (1 - \beta) ((\mathbf{u}_{h,\mathfrak{A}} \otimes \mathbf{n}_\Gamma) : \boldsymbol{\sigma}_{h,\mathfrak{A}} - (\mathbf{u}_{h,\mathfrak{B}} \otimes \mathbf{n}_\Gamma) : \boldsymbol{\sigma}_{h,\mathfrak{B}}) \, dS. \end{aligned} \quad (5.38)$$

We also have to add the discretization of the surface stress force presented in Sec. 2.6 to the right hand side of the discretization (Eq. 4.67), such that:

$$r_2^{\text{XDG}}(\mathbf{v}_h) = r_2(\mathbf{v}_h) + r_F(\mathbf{v}_h). \quad (5.39)$$

In case of a constant surface stress as shown in Eq: (2.78) the source term is:

$$r_F(\mathbf{v}_h) = \oint_{\mathcal{J}} \frac{1}{\text{We}} \kappa \mathbf{n}_{\mathcal{J}} \cdot \llbracket \mathbf{v}_h \rrbracket \, dS \quad (5.40)$$

The mean curvature κ can be calculated from the level-set using Bonnets formula:

$$\kappa := \nabla \cdot \mathbf{n}_{\mathcal{J}} = \nabla \cdot \left(\frac{\nabla \varphi}{|\nabla \varphi|} \right). \quad (5.41)$$

The main issue using Bonnet's formula for the evaluation of the isotropic stress tensor is its extreme sensitivity to minor disturbances in the level-set function φ such as jumps in the derivatives or the non-exact representation of the signed-distance function in a DG context. One possibility used in this work is using a formulation of the isotropic part of the surface

stress tensor circumventing the evaluation of the curvature. This formulation is called the Laplace-Beltrami approximation and reads:

$$\oint_{\mathfrak{J}} \sigma \boldsymbol{\kappa} \cdot \mathbf{v} \, dS = \oint_{\mathfrak{J}} \sigma \mathbf{P}_{\mathfrak{J}} : \nabla \mathbf{v} \, dS - \int_{\mathfrak{J} \cap \Gamma_I} \sigma \mathbf{z} \cdot \mathbf{v} \, dl_s \quad (5.42)$$

where \mathbf{z} is tangential to the interface \mathfrak{J} , means $\mathbf{z} \cdot \mathbf{n}_{\mathfrak{J}} = 0$, and normal onto $\partial(\mathfrak{J} \cup K_j)$ and $\int \dots \, dl_s$ denotes an integral over the boundary of the surface $\mathfrak{J} \cup K_j$. In two dimensions this means that it is a zero-dimensional point integral over the two endpoints of the line $\mathfrak{J} \cup K_j$. The main benefit of this reformulation is that it is not dependent on second order derivatives of the level-set φ .

The Boussinesq-Scriven model for the surface stress force is discretized as follows:

$$\begin{aligned} r_F(\mathbf{v}_h) = & \frac{1}{\hat{\rho} u_c^2 l_c} \left(- \oint_{\mathfrak{J}} \sigma \mathbf{P}_{\mathfrak{J}} : \nabla_{\mathfrak{J}} \mathbf{v}_h \, dS + \int_{\mathfrak{J} \cap \Gamma_I} \sigma \{\mathbf{z}\} \cdot \llbracket \mathbf{v}_h \rrbracket \, dl_s \right. \\ & - \oint_{\mathfrak{J}} \tilde{\Lambda}_{\mathfrak{J}} (\nabla_{\mathfrak{J}} \cdot \mathbf{u}) \mathbf{P}_{\mathfrak{J}} : \nabla_{\mathfrak{J}} \mathbf{v}_h \, dS + \int_{\mathfrak{J} \cap \Gamma_I} \{ \tilde{\Lambda}_{\mathfrak{J}} (\nabla_{\mathfrak{J}} \cdot \mathbf{u}) \mathbf{z} \} \cdot \llbracket \mathbf{v}_h \rrbracket \, dl_s \\ & - \oint_{\mathfrak{J}} \eta_{\mathfrak{J}} (\nabla_{\mathfrak{J}} \mathbf{u} + (\nabla_{\mathfrak{J}} \mathbf{u})^T) \mathbf{P}_{\mathfrak{J}} : \nabla_{\mathfrak{J}} \mathbf{v}_h \, dS \\ & \left. + \int_{\mathfrak{J} \cap \Gamma_I} \{ \eta_{\mathfrak{J}} (\nabla_{\mathfrak{J}} \mathbf{u} + (\nabla_{\mathfrak{J}} \mathbf{u})^T) \mathbf{z} \} \cdot \llbracket \mathbf{v}_h \rrbracket \, dl_s \right). \end{aligned} \quad (5.43)$$

6. Solution Strategies for the Algebraic Equation System

In this chapter we give an overview over several strategies which have been identified to improve the convergence of the solution algorithm for the mixed elliptic-hyperbolic system of equations for steady flows. After we first emphasize the structure of the solver and the interaction of the different strategies, we then briefly describe the Newton method used in the solver for linearization. It has been implemented in BoSSS in the working group parallel to the viscoelastic implementation and it has massively improved the convergence properties of the solution of this system of equations.

In the last section some strategies implemented by the author for improving the convergence performance are discussed. Up to now, in the current two-phase flow test cases, there is no necessity dealing with troubled cells since there are no steep gradients or boundary layers. However, in case of multi-phase flow most issues are concerning the level-set and its continuity across the edges of discontinuous elements, for which there are other strategies such as adaptive mesh refinement along the interface which has been presented in Chap. 5.

Most parts of this chapter are deduced from a submitted publication currently under review (Kikker et al., 2020).

6.1. Solver Structure

The presented solver, which is exclusively using a DG scheme in the context of viscoelastic flow, is implemented in the software framework BoSSS Kummer (2012) and due to its application in multi-phase flows embedded in the eXtended Discontinuous Galerkin (XDG) solver presented by Kummer (2017). The overall solver consist of several levels with different iteration schemes:

1. time-stepping scheme,
2. homotopy scheme with slowly increasing Weissenberg number,
3. troubled cell treatment with different artificial viscosities or adaptive mesh refinement,
4. level-set movement
5. Newton scheme using a direct solver.

The overall algorithm within one time-step (1) is shown in a flowchart in Fig. 6.1. The item 'solve' contains in case of multi-phase flow in a first step the level (4) which is the re-initialisation, the evaluation of the extensional velocity, the actual movement of the level-set, and the determination of the agglomerated and adaptively curvature refined grid, and it contains in a second step the level (5) which is the Newton scheme and a direct solver. In case of single-phase flow the level (4) is skipped or filled with dummy variables having a stationary level-set outside the domain, respectively.

For the linearisation we use a Newton method (Kelley, 1995) and to solve the linear system, we use a third party direct solver.

Since with increasing Weissenberg number the solver becomes more unstable and finding a convergent solution without an adequate initial guess becomes difficult, we use a homotopy method starting with a Newtonian solution and slowly increasing Weissenberg number within the time-step for steady calculations (level (2)). Therefore, we have an aim Weissenberg number and an increment of it which is chosen to be $Wi_{\text{incr}} = 0.1$. This value has proven to be small enough for a good convergence for the Newton method without causing too many additional iteration cycles increasing the computational time. The level (3) is described in detail in section 6.3.

6.2. Linearization of the System with the Newton method

When the implementation of the system of equations for viscoelastic flow started, the BoSSS framework used for the iterative solution of the non-linear system a simple fix-point iteration, also called Picard iteration, with a non-linear fix-point map. Although this method is cheap and robust and apparently working for the saddle-point problem of Newtonian Navier-Stokes flow, it does not converge for the extended non-saddle-point system of equations for viscoelastic flow. Thus, the Newton method has been implemented in BoSSS for the viscoelastic flow solver. Additionally, it reduces computing time in all applications in BoSSS, e.g. three-dimensional flow, because of the second order convergence behaviour of the method. The information in this section about the Newton method are mainly deduced from Kelley (1995).

From the discretization we obtain a non-linear system of equations to solve which can be displayed as follows:

$$\mathcal{A}(\mathbf{U}) = 0 \quad (6.1)$$

with \mathcal{A} is the non-linear operator depending on the solution vector \mathbf{U} . We define the Jacobian Matrix:

$$\mathcal{J}_{ij}(\mathbf{U}) := \frac{\partial \mathcal{A}_i}{\partial U_j}(\mathbf{U}). \quad (6.2)$$

With this, we can define for all $\mathbf{U} \in \Omega$ sufficiently close to the solution \mathbf{U}^* :

$$\mathcal{A}(\mathbf{U}) - \mathcal{A}(\mathbf{U}^*) = \int_0^1 \mathcal{J}(\mathbf{U}^* + s(\mathbf{U} - \mathbf{U}^*)) (\mathbf{U} - \mathbf{U}^*) \, ds. \quad (6.3)$$

An iterative method for computing \mathbf{U} is locally convergent if the iterates converge to \mathbf{U}^* providing that the initial data is sufficiently good. The fact that convergence becomes difficult, if the initial guess is too far away from the solution is an issue for viscoelastic flow problems and will be discussed in the next Sec. 6.2.3.

Using the Newton method to solve our system of equations we must make some assumptions:

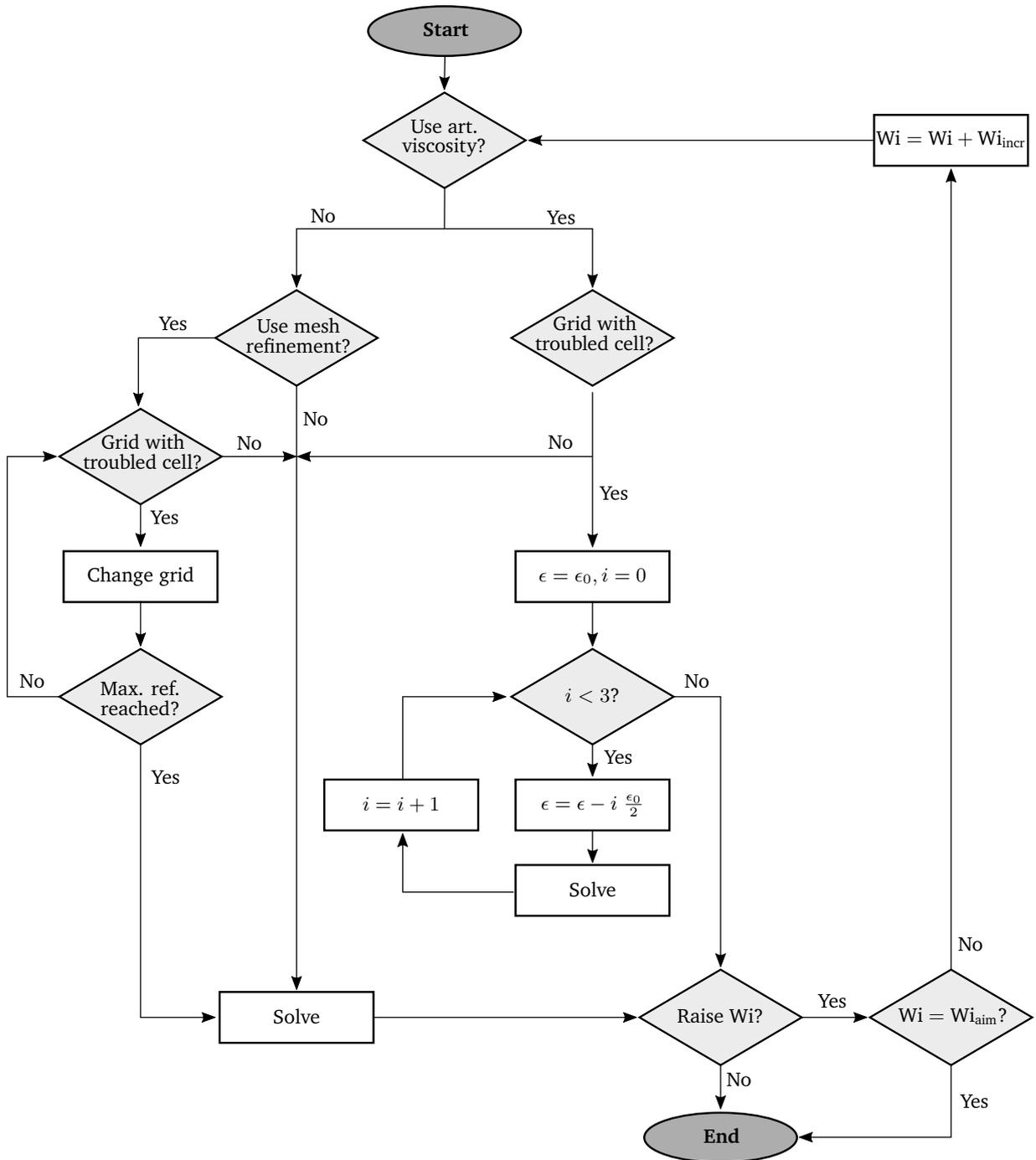


Figure 6.1.: Flowchart of the solver within a steady or the first time-step with different levels of iterative loops. The item 'solve' contains the linearisation scheme and the direct solver for the linearised system. In case of multi-phase flow it also includes the movement of the level-set. ϵ is the maximum artificial viscosity used and ϵ_0 the user defined starting value for ϵ . In unsteady simulations the Weissenberg number is fixed after the first time-step with Wi_{aim} .

- $\mathcal{A}(\mathbf{U}) = 0$ has a solution \mathbf{U}^* .
- \mathcal{J} is Lipschitz continuous on Ω with Lipschitz constant ι because $\|\mathcal{J}(\mathbf{U}) - \mathcal{J}(\mathbf{U}^*)\| \leq \iota \|\mathbf{U} - \mathbf{U}^*\|$.
- $\mathcal{J}(\mathbf{U}^*)$ is nonsingular, so the condition number is not ∞ .

6.2.1. Implementation of the Newton Method

In this work, the non-linear matrix $\mathcal{A}(\mathbf{U})$ has a special structure presented in Eq. (4.27), such that

$$\begin{aligned} \mathcal{A}(\mathbf{U}) &= A_U(\mathbf{U})\mathbf{U} - \mathbf{b}, \\ \mathcal{J}(\mathbf{U})_{ij} &= A_{Uij} + \sum_k \frac{\partial A_{Uik}}{\partial U_j} U_k = A_{Uij} + A_{U'ij} U_k. \end{aligned} \quad (6.4)$$

Now we can construct the Newton method. We consider \mathbf{U}_{n+1} being the root of the two-term Taylor expansion $M_n(\mathbf{U})$ of \mathcal{A} around \mathbf{U}_n :

$$M_n(\mathbf{U}) := \mathcal{A}(\mathbf{U}_n) + \mathcal{J}(\mathbf{U}_n)(\mathbf{U} - \mathbf{U}_n), \quad (6.5)$$

leading to:

$$\mathbf{U}_{n+1} = \mathbf{U}_n - \mathcal{J}(\mathbf{U}_n)^{-1} \mathcal{A}(\mathbf{U}_n) \quad (6.6)$$

and the Newton step from the current iterative \mathbf{U}_n to the new iterative \mathbf{U}_{n+1} is

$$\mathbf{s} := -\mathcal{J}(\mathbf{U}_n)^{-1} \mathcal{A}(\mathbf{U}_n) = \mathbf{U}_{n+1} - \mathbf{U}_n. \quad (6.7)$$

To find an adequate termination criterion we define the relative non-linear residual with

$$R := \frac{\|\mathcal{A}(\mathbf{U})\|}{\|\mathcal{A}(\mathbf{U}_0)\|} \quad (6.8)$$

which is related to the error, defined as $\mathbf{e} := \mathbf{U} - \mathbf{U}^*$ as follows:

$$\frac{\|\mathbf{e}\|}{4\|\mathbf{e}_0\|\text{cond}(\mathcal{J}(\mathbf{U}^*))} \leq R \leq \left(\frac{\|\mathbf{e}_0\|}{4\|\mathbf{e}\|\text{cond}(\mathcal{J}(\mathbf{U}^*))} \right)^{-1} \quad (6.9)$$

with $\text{cond}(\mathcal{J}(\mathbf{U}^*))$ the condition number of the Jacobian of \mathbf{U}^* . If the condition number is small, the size of the relative non-linear residual R is a good indicator for the size of the error. So in our implementation we use the termination criterion:

$$\|\mathcal{A}(\mathbf{U})\| \leq \text{tol}_r \cdot \|\mathcal{A}(\mathbf{U}_0)\| + \text{tol}_a \quad (6.10)$$

where we choose the relative and the absolute error tolerance (tol_r and tol_a , respectively) to be the same. The idea of having a relative and an absolute error tolerance is the following: If there is an error in the evaluation of the matrix \mathcal{A} or the initial guess is close to a root, a termination criterion solely based on the relative residual may be made too late in the iteration or the iteration may not terminate at all. Additionally, we have no right-hand side in the general form of Eq. (6.1) which can be used as a scaling factor like it is the case in a classical linear system. So the relative and absolute residual must be balanced differently. In

the implementation in BoSSS we choose the same value for the absolute and relative error tolerance. More information about the termination criterion and different other choices as well as the deduction of Eq. (6.9) can be found in Kelley (1995). Considering the error of the current and the new iterate it can be deduced that

$$\|\mathbf{e}_{n+1}\| \leq k \cdot \|\mathbf{e}_n\|^2 \quad (6.11)$$

so the the Newton method converges quadratically in the error to the solution \mathbf{U}^* . More information about the convergence of the Newton method can also be found in Kelley (1995). The pseudocode of the implementation of the Newton method is presented in Alg. 1. For the solution of the linearized system of equations we use a direct solver, namely the direct solver framework PARDISO (Parallel Sparse Direct and Multi-Recursive Iterative Linear Solvers) from the Intel(R) Parallel Studio XE 2018, Update 3, Cluster Edition for Windows (Schenk and Gärtner, 2002) or MUMPS (Multi-frontal Massively Parallel Solver), Version 5.1.2 (Agullo et al., 2017). However, since the amount of governing equations and dependent variables is doubled in case of two-dimensional flow compared to a discretization of a Navier-Stokes system, the direct solvers have computational and memory issues for the increased size of operator matrix. To circumvent this, there are preconditioning schemes for high performance computing such as multigrid methods (p-multigrid) or additive Schwarz and orthonormalization schemes in combination with iterative linear solvers like GMRES currently under development within the BoSSS framework. For more information about these solvers the reader is referred to Kummer et al. (2020b).

In order to compute the Newton iterate \mathbf{U}_{n+1} from a current solution \mathbf{U}_n one must first evaluate the action of the operator matrix on \mathbf{U}_n , which is $\mathcal{A}(\mathbf{U}_n)$, and decide whether to terminate the iteration. Next, the Jacobian $\mathcal{J}(\mathbf{U}_n)$ is computed and factorized. With this, the Newton step is evaluated as a solution of $\mathcal{J}(\mathbf{U}_n)\mathbf{s} = -\mathcal{A}(\mathbf{U}_n)$ (Eq. (6.7)) and the iteration is updated to $\mathbf{U}_{n+1} = \mathbf{U}_n + \mathbf{s}$ (Kelley, 1995). Before the linear system of the Newton step is given to the direct solvers, the matrices are preconditioned using a block preconditioner with the possibility to choose different preconditioner for different blocks of the matrix, which is to reduce the condition number of the system and therefore, reduce the numerical error. For the factorization of the matrices both direct solvers use a LU-factorization.

Algorithm 1: Newton method

Input: initial iterate \mathbf{U}_0 , non-linear map \mathcal{A} , termination tolerance tol

Output: approximate solution \mathbf{U}

```

1  $R_0 = \|\mathcal{A}(\mathbf{U})\|$ 
  // Newton iteration...
2 while  $\|\mathcal{A}(\mathbf{U})\| > tol \cdot R_0 + tol$  do
3   |   Compute  $\mathcal{J}(\mathbf{U})$ 
4   |   Factor  $\mathcal{J}(\mathbf{U}) = LU$ 
   |   // Newton step...
5   |   Solve  $LU\mathbf{s} = -\mathcal{A}(\mathbf{U})$ 
6   |    $\mathbf{U} = \mathbf{U} + \mathbf{s}$ 
7   |   Evaluate  $\mathcal{A}(\mathbf{U})$ 

```

6.2.2. Choice of the Approximation of the Jacobian

During development of the Newton method in the solver framework BoSSS three different approaches for the approximation of the Jacobian in Eq. (6.2) or Eq. (6.4), respectively, were followed. The first choice was an ad-hoc linearisation for the Jacobian, the second choice was the use of an explicit finite-difference approximation of the Jacobian. Both showed advantages and disadvantages in the implementation. Finally, the Jacobian derivatives were included in the fluxes of the discretization leading to a parameter-free discretization of the system. If we insert relations (6.4) into the Newton iterative (6.7) and rearrange the terms, we get:

$$A_U(\mathbf{U}_n)\mathbf{U}_{n+1} - \mathbf{b} + A'(\mathbf{U}_n)\mathbf{U}_n\mathbf{s} = 0. \quad (6.12)$$

In this notation we see that the Newton iteration divides into a fix-point or Picard system and a first order part where the Jacobian part A' has to be approximated. If it is set to zero, we obtain the fix-point linearization. The three different variants of approximation and their advantages and disadvantages are explained in the following.

Ad-hoc Linearization

In case of saddle-point problems, which can also be solved using fix-point iteration, the Jacobian of A_U which is A' is very similar to the operator matrix itself. So the Jacobian can be approximated using the operator matrix itself and we get for the Newton iteration:

$$A_U(\mathbf{U}_n)\mathbf{U}_{n+1} - \mathbf{b} + A_U(\mathbf{U}_n)\mathbf{U}_n\mathbf{s} = 0. \quad (6.13)$$

This approximation of the Jacobian is cheap in terms of computational costs and uses only local dependencies. Thus, it omits derivative entries in secondary diagonals of the Jacobian which are negligible in case of a saddle-point problem. However, since we do not have a saddle-point problem in the operator matrix in case of viscoelastic flow problems, this method fails to converge like the fix-point approach.

Finite-Differences Jacobian

In case of the finite differences approximation of the Jacobian we assume that we compute $A_U(\mathbf{U}_n + \varepsilon(\mathbf{U}_n))$ instead of $A(\mathbf{U}_n)$ and the Matrix $A'(\mathbf{U}_n)$ is calculated by the approximate action of it on a vector using forward differencing. We choose $\varepsilon = \varepsilon\|\mathbf{U}_n\|\mathbf{e}_j$ where ε should be $\varepsilon = \hat{\varepsilon}^{\frac{1}{2}}$ to minimize the error. Here, $\hat{\varepsilon} \approx 10^{-15}$ is the floating point round-off in single precision. So $\varepsilon \approx 10^{-7}$ is a good choice. It should not be chosen too small since this can lead to highly inaccurate results. Hence, in the BoSSS framework, the root of the floating point round-off is calculated and chosen for ε . Using now the forward differencing the Jacobian is approximated with (Kelley, 1995):

$$\tilde{A}'(\mathbf{U}_n)_j := \frac{A_U(\mathbf{U}_n + \varepsilon\|\mathbf{U}_n\|\mathbf{e}_j) - A_U(\mathbf{U}_n)}{\varepsilon\|\mathbf{U}_n\|} \quad (6.14)$$

and the Newton iteration reads:

$$A_U(\mathbf{U}_n)\mathbf{U}_{n+1} - \mathbf{b} + \tilde{A}'(\mathbf{U}_n)\mathbf{U}_n\mathbf{s} = 0. \quad (6.15)$$

This approximation has the advantage to display also non-local dependencies and dependencies in the parameters such as a flux formulation of the gradient of the parametric velocity, which also leads to better convergence properties for non-saddle-point problems. However, the computational effort is very high. If A_U is an $N \times N$ matrix, the computation of the finite differences approximation of A' is N times the evaluation of A_U since every computation of a column of \tilde{A}' requires the evaluation of A_U .

Parameter-Free Jacobian

In case of the parameter-free linearization and calculation of the Jacobian the values of \mathbf{U}_n are not stored separately and used for an explicit evaluation of the matrices. Instead, the linearization point is taken implicitly with \mathbf{U}_{n+1} and the derivatives for the Jacobian A' are integrated into the weak formulation and the fluxes of the system of equation. So the Newton iteration is:

$$A_U(\mathbf{U}_{n+1})\mathbf{U}_{n+1} - \mathbf{b} + A'(\mathbf{U}_{n+1})\mathbf{U}_n\mathbf{s} = 0. \quad (6.16)$$

This implementation combines the advantageous properties of the both strategies above. While it is as fast as the first strategy it can also display non-local dependencies, but no dependencies in the parameters.

6.2.3. Incremental Increase of the Weissenberg Number

As already mentioned in Sec. 6.2 it is very important for the Newton method to use an appropriate initial guess for the first iterate \mathbf{U}_0 to succeed in a quadratic order convergence to the solution. This is due to the fact that all the theory about the Newton method holds under the assumption that there is a $\delta > 0$ such that for all $\mathbf{U} \in B(\delta)$

$$\begin{aligned} \|\mathcal{J}(\mathbf{U})\| &\leq 2\|\mathcal{J}(\mathbf{U}^*)\| \\ \|\mathcal{J}(\mathbf{U})^{-1}\| &\leq 2\|\mathcal{J}(\mathbf{U}^*)^{-1}\| \quad \text{and} \\ \|\mathcal{J}(\mathbf{U}^*)^{-1}\|^{-1} \frac{\|\mathbf{e}\|}{2} &\leq \|\mathcal{J}(\mathbf{U})\| \leq 2\|\mathcal{J}(\mathbf{U}^*)\| \|\mathbf{e}\| \end{aligned} \quad (6.17)$$

Here, the solution of the system is \mathbf{U}^* and the ball of radius about \mathbf{U}^* is defined as

$$B(\delta) := \{\mathbf{U} \mid \|\mathbf{e}\| < \delta\} \quad (6.18)$$

with the error $\mathbf{e} = \mathbf{U} - \mathbf{U}^*$ defined before. This means, that we must ensure to have an initial iterate being located within the ball of radius δ (Kelley, 1995). In many cases, it is sufficient to choose $\mathbf{U}_0 = \mathbf{0}$, but in the case of highly elastic viscoelastic flow with $Wi \gg 0$, the initial guess must be located closer to the solution \mathbf{U}^* . We reach this using a homotopy continuation method (e.g. Wu, 2005) slowly increasing the Weissenberg number from Newtonian flow in the first time-step as can be seen in the flowchart (Fig. 6.1). Therefore, we start with $\mathbf{U}_0 = \mathbf{0}$ or an IC for Newtonian flow and $Wi = 0$ adding an increment of the Weissenberg number to it in the operator matrix after the Newton method found a convergent solution. This strategy is followed until the aim Weissenberg number is reached.

6.3. Convergence Supporting Strategies

As already discussed at many points in this work reaching a convergent solution for viscoelastic fluid flow is particularly challenging because of physical instabilities, the difficulty of well-posedness of the system and the mixed elliptic-hyperbolic type of equation system. So beside using a robust and quadratic convergent non-linear solver, namely the Newton method, some different strategies for helping the solver to find a convergent solution were developed. They are presented and discussed in this section.

As mentioned in Sec. 1, solving viscoelastic flow problems such as the confined cylinder benchmark presented in this work can cause convergence problems due to steep velocity and stress gradients especially at the cusp and in the wake of the cylinder. This is particularly the case for high order DG methods where less numerical diffusion occurs. To overcome these problems we employ a standard method initially designed for shock-capturing in compressible flow using an artificial diffusion. The artificial diffusion smooths the steep gradients into the neighbouring elements until a convergent solution is reached. The artificial viscosity is then slowly reduced and eliminated such that there is no more diffusion in the final solution. For the high-order DG approach we follow the artificial viscosity based shock-capturing approach introduced by Persson and Peraire (2006). They introduce a dependency of the artificial viscosity not only on h but on $\frac{h}{k}$, which is the resolution given by a piecewise polynomial of order k , since sub-cell resolution is possible. This means that for a fixed grid resolution h the steep gradient profiles can become thinner than the element size with increasing k .

The artificial diffusion is added to the constitutive equations with a SIP-discretized Laplacian of the stresses and multiplied with an artificial viscosity field ϵ_h .

6.3.1. Troubled Cell Indicator

The solution in each element is written as a group of orthogonal polynomials. In the case of a solution with small gradients the coefficients are expected to decay very quickly. On the other hand, the coefficients will have a low rate of decay for solutions with high gradients. We introduce a truncated solution \hat{u}_h on the same basis as the solution u_h but with order $(k - 1)$ with

$$\langle u_h - \hat{u}_h, v_h \rangle = 0 \quad \forall v_h \in \mathbb{V}_k. \quad (6.19)$$

Using these solutions, Persson and Peraire now define a cell-local smoothness indicator (Persson and Peraire, 2006):

$$S_h := \frac{\langle u_h - \hat{u}_h, u_h - \hat{u}_h \rangle}{\langle u_h, u_h \rangle}. \quad (6.20)$$

If we have a smooth and at least continuous solution, we expect the coefficients of the polynomial expansion and thus, the sensor S_h to decay at least with $\sim 1/k^4$, which Persson and Peraire (2006) confirmed with numerical results. As a sensing parameter we can e.g. consider the normal component of the stress tensor τ_{xx} since steep gradients are often expected in the x -component of the velocity in x -direction. In other words, due to the steep gradients the normal stress τ_{xx} can quickly reach very high values.

6.3.2. Artificial Diffusion

If the smoothness indicator identifies a cell with discontinuity, the artificial viscosity must be set element-wise depending on the steepness of the gradients. Persson and Peraire use a smooth Heaviside function approach (Persson and Peraire, 2006):

$$\epsilon_h := \begin{cases} 0, & \text{if } s_h < s_0 - \mu, \\ \frac{\epsilon_0}{2} \left(1 + \sin \left(\frac{\pi(s_h - s_0)}{2\mu} \right) \right), & \text{if } s_0 - \mu \leq s_h \leq s_0 + \mu, \\ \epsilon_0, & \text{if } s_h > s_0 + \mu \end{cases} \quad (6.21)$$

with $s_h := \log_{10}(S_h)$. In our case we add scaling factors to the maximum artificial viscosity value $\epsilon_0 \sim \frac{h}{k}$ and the user-defined critical sensor value $s_0 \sim \log_{10}(\frac{1}{k^4})$. The bound $\mu \sim \mathcal{O}(1)$ has to be chosen such that the solution is sufficiently smoothed at discontinuities. The algorithm written in pseudocode is presented in 2. In Fig. 6.2 the sensor field deduced from the normal stress field and the corresponding artificial viscosity field are shown.

Algorithm 2: Calculating the artificial viscosity for troubled cells

Input: sensor field, degree k of tested field, upper bound, maximum viscosity ϵ_0

Output: artificial viscosity field ϵ_h

```

1 sensor value = log10 (sensor field)
2 limit = log10 (upper bound / k4)
  // Go over all cells...
3 foreach cell  $K \in \mathfrak{K}$  do
  | // set artificial viscosity in the cell
4   if sensor value( $K$ ) < limit - 1 then
5     |  $\epsilon_h(K) = 0$ 
6   else if sensor value( $K$ ) > limit + 1 then
7     |  $\epsilon_h(K) = \epsilon_0$ 
8   else
9     |  $\epsilon_h(K) = \frac{1}{2}\epsilon_0 (1 + \sin(\frac{1}{2}\pi(\text{sensor value} - \text{limit})))$ 

```

6.3.3. Adaptive Mesh Refinement for Troubled Cells

A second strategy to treat troubled cells found with the indicator in Eq. (6.20) is to apply an adaptive mesh refinement within these cells. This method is used for Cartesian grids. After a troubled cell is identified by the sensor value exceeding a user defined threshold, the refinement level of this cell is increased and it is divided in four cells of the same size (Alg. 3). Afterwards, the neighboring cells are refined, if necessary, such that one edge does not neighbour four edges. In the next step the troubled cell indicator is recalculated and troubled cells are refined again until a user-defined refinement level is reached. Contrary, if a cell is identified where the sensor value fall below a lower threshold, it is approved to be coarsened if it is allowed by the neighbouring cells until the coarseness of the starting grid is reached. An example for the refinement strategy is illustrated in Fig. 6.3 for the steady calculation of a

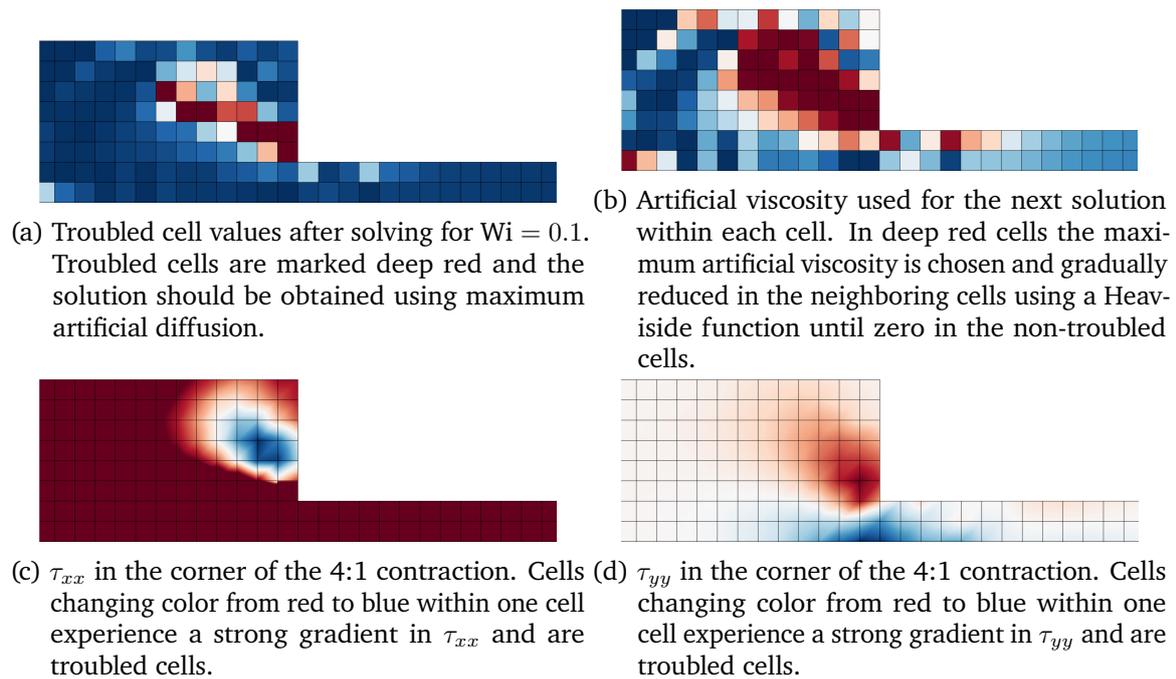


Figure 6.2.: Artificial diffusion using the troubled cell indicator presented for the refinement strategy. Since in this representation only qualitative values of the quantities are of interest, scale bars are omitted for simplification. Red colored cells mark high values, blue colored cells mark low values. in case of the troubled cell indicator the upper and lower bound s_0 are also used in the scaling of the coloring such that deep red and deep blue cells respectively, show cells with a troubled cell indicator out of the bounds s_0 .

4:1 contraction for an increase of the Weissenberg number from 0 to 0.1. If turned on, this algorithm is repeated whenever the system is solved with an increased Weissenberg number or after a new time-step in unsteady calculation.

Though this algorithm improves the balance between intense refinement of the mesh where most changes in the flow are expected without over-refinement in not useful areas with corresponding increased computational costs, it shows a disadvantage in conjunction with viscoelastic flow. Especially in benchmark problems which are widely discussed in literature concerning the high Weissenberg number problem (covered in Chap. 3), steep velocity gradients cause very thin boundary layers and exponential growth of the stresses (e.g. Claus and Phillips, 2013; Dou and Phan-Thien, 2007; Fattal and Kupferman, 2005). Convergence is often only reached if the meshes used under-resolve these gradients and boundary layers such that small structures within one element are not detected. So using adaptive mesh refinement in exactly these regions namely increases the accuracy to resolve these structures, but enhances a loss of convergence finding an appropriate solution.

Algorithm 3: Applying adaptive mesh refinement on troubled cells

Input: list of cells \mathcal{K} , sensor field, desired refinement level, upper bound, factor for lower bound

Output: adapted mesh

// Go over all cells...

```
1 foreach cell  $K \in \mathcal{K}$  do
  // create list of cells for refinement
2  if  $sensor(K) > upper\ bound$  then
3    if  $refinement\ level\ of\ K < desired\ refinement\ level$  then
4       $\lfloor$  add  $K$  to list of cells for refinement
  // create list of cells blocked for coarsening
5  lower bound = upper bound  $\cdot$  factor for lower bound
6  if  $sensor(K) \leq upper\ bound$  then
7    if  $sensor(K) > lower\ bound$  then
8       $\lfloor$  add  $K$  to list of cells for not to coarsen
9 Adapt mesh regarding list of cells for refinement and list of cells not to coarsen
```

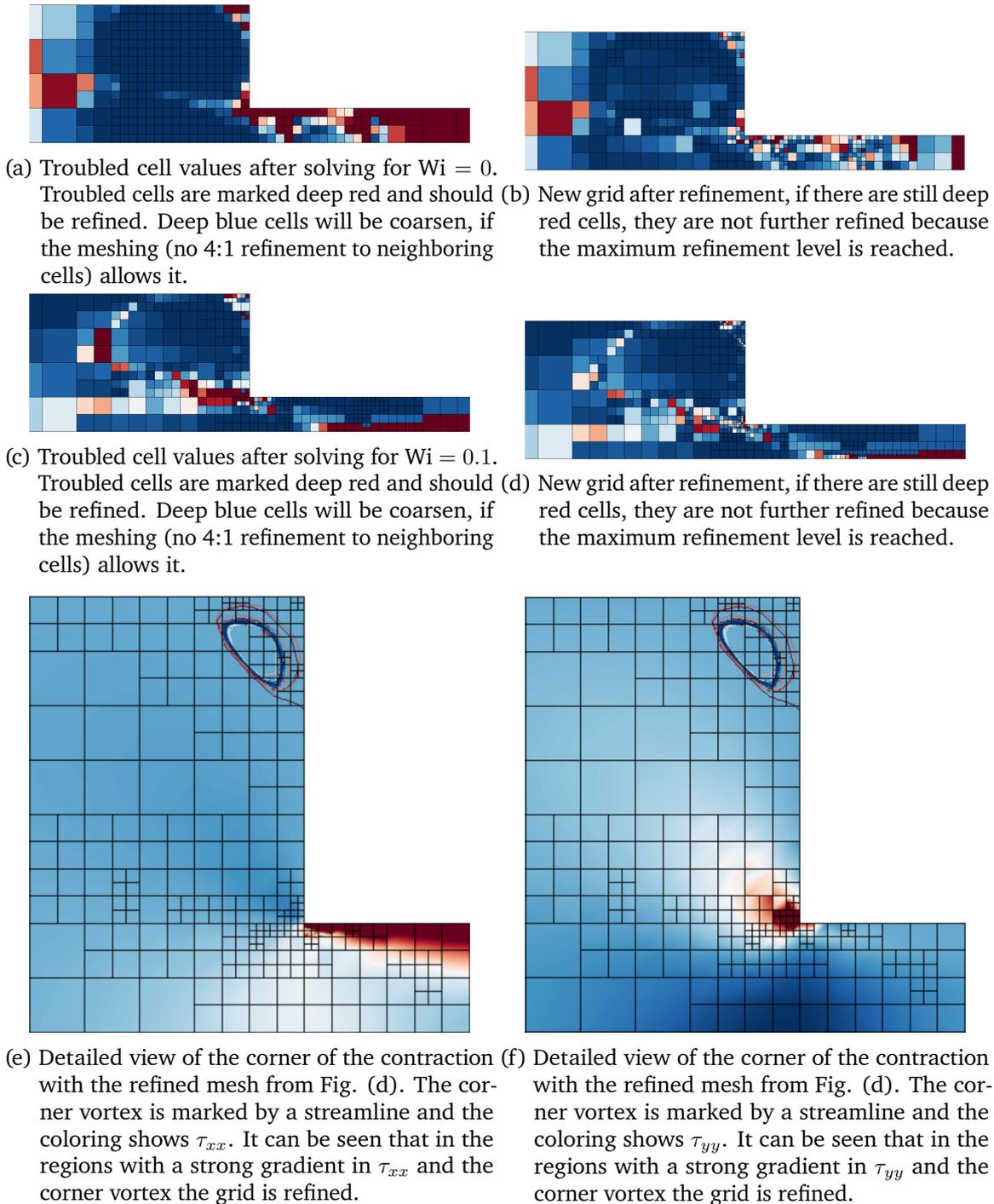


Figure 6.3.: Adaptive mesh refinement using the troubled cell indicator presented for the refinement strategy. Since in this representation only qualitative values of the quantities are of interest, scale bars are omitted for simplification. Red colored cells mark high values, blue colored cells mark low values.

7. Numerical Results

In this chapter we present numerical results conducted by the `Rheology_Solver` and the `XRheology_Solver`. The former was used to verify and validate the implementation of the fluxes for the governing viscoelastic system of equations for the single-phase case. The results presented here are a convergence study for a non-polynomial manufactured solution using the LDG discretization, and a more detailed study on the confined cylinder benchmark problem where the results were mainly compared to the literature.

For the multi-phase purpose the `XRheology_Solver` was used, which is a combination of the first with the multiphase XDG solver (`XNSE_Solver`) within the `BoSSS` code. Since at this point some detailed developmental work solving some issues concerning the behaviour of the fluxes at the interface is still necessary, the results in the last section have to be seen as a feasibility study for viscoelastic two-phase flow and not as a quantitative analysis. The template for this study is a droplet in shear flow setting analysed by Chinyoka et al. (2005).

7.1. Verification of the Local Discontinuous Galerkin Method

The results presented here verify the implementation of the LDG method using a manufactured solution and to show the capability of the method to solve mixed hyperbolic-elliptic or hyperbolic systems of PDE for the steady case. The section is based on a published work of the author presented in Kikker and Kummer (2018). Contrary to that work we use here the full Navier-Stokes system with $Re \neq 0$ instead of the Stokes system presented in that proceeding. The manufactured solution is the same as in Cockburn et al., 2002 to which the method was compared. We use the discretization of system (2.66)-(2.68) with the material parameters: $Re = 0, 1$, $Wi = 0$ and $\beta = 0$ which leads to a Newtonian flow in the inertia-less Stokes system ($Re = 0$) and the Navier-Stokes system ($Re = 1$), but in both cases with two first order PDE making the system hyperbolic ($\beta = 0$) without an elliptic second-order fraction in the momentum equation (2.67).

We use the manufactured non-polynomial divergence free solution introduced by Cockburn et al. (2002):

$$\begin{aligned}u_x(x, y) &= -e^x (y \cos(y) + \sin(y)), \\u_y(x, y) &= e^x y \sin(y), \\p(x, y) &= 2e^x \sin(y).\end{aligned}\tag{7.1}$$

The domain is $\Omega = (-1, 1)^2$. Appropriate source terms on the right-hand side are added to satisfy the solution. The mesh of the domain is uniform with quadratic elements of size $h = 2^{-n+1}$ where h is the grid spacing and n is the grid level. The manufactured solution

Table 7.1.: EOC of Cockburn et al. (2002) (lit.) and of our LDG solver (here) for the Stokes system.

k	$\ \mathbf{u} - \mathbf{u}_{ex}\ _{L^2}$		$\ p - p_{ex}\ _{L^2}$		$\ \boldsymbol{\tau} - \boldsymbol{\tau}_{ex}\ _{L^2}$	
	lit.	here	lit.	here	lit.	here
1	2.04	1.18	1.36	0.91	0.82	0.90
2	3.05	3.44	2.18	2.88	1.85	2.11
3	4.06	4.10	2.77	3.27	2.80	3.04
4	-	5.05	-	4.09	-	3.92

(7.1) is non-polynomial such that it cannot be displayed exactly by the test functions v_h . For all boundaries a velocity inlet BCn is chosen.

First, we briefly present the results for the Stokes system for $Re = 0$ found in Kikker and Kummer (2018). The grid convergence is shown in Tab. 7.1 for different polynomial orders and compared to the results of Cockburn et al. (2002). They are in good agreement with the predicted order of convergence of $k + 1$ for the velocity and k for the pressure and the stresses, respectively, whereas k is the polynomial order of the test and trial functions. For first order polynomials as test functions we do not reach second order convergence for the velocity. For all other polynomial orders we reach higher convergence orders than predicted and than presented in Cockburn et al. (2002).

With the improved version of the code we revise the convergence study for the Navier-Stokes system ($Re = 1$). In this case the momentum transfer presented in Eq. (3.11) can easily evoke a change of type for viscoelastic fluids to hyperbolic, as illustrated in Chap. 3. So we want to ensure that the hyperbolic system of equations for the limiting case of a Newtonian flow ($Wi = 0$) is stable without adding numerical diffusion in the convective term, thus, we examine if it converges and the solution is consistent. The results are presented in Fig. 7.1 and the EOC can be found in Tab. 7.2 up to a polynomial order of $k = 5$ for the velocity and stresses ($k - 1$ for the pressure). They are all in good agreement with the predicted order of convergence of $k + 1$ for the velocity and k for the pressure and the stresses, respectively.

For the proof of stability of the discretization of the whole system of equations (2.66)-(2.68) with $Wi \neq 0$ and $\beta \neq 0$ the reader is referred to Sec. 7.2 where validating results for the confined cylinder benchmark problem and also a convergence study for that test-case can be found.

7.2. Validation by the Confined Cylinder Benchmark Problem

Since only few analytical solutions exist for complex viscoelastic flow, several benchmark problems have been established for validating numerical schemes and models. One of these is the confined cylinder problem in which the flow around a cylinder immersed in a narrow channel with a blocking ratio of 2:1 is investigated numerically. This benchmark has been analysed in numerous works, e.g. Baaijens et al. (1997), Fan et al. (2005), Oliveira and Miranda (2005), Dou and Phan-Thien (2007), Bonito and Burman (2008), Claus and Phillips (2013), and Keith et al. (2017), but still, the numerical treatment of this testcase is challenging

Table 7.2.: EOC of our LDG solver for the Navier-Stokes system with $\text{Re} = 1$ using the manufactured solution in Eq. (7.1).

k	$\ \mathbf{u} - \mathbf{u}_{ex}\ _{L^2}$	$\ p - p_{ex}\ _{L^2}$	$\ \boldsymbol{\tau} - \boldsymbol{\tau}_{ex}\ _{L^2}$
1	1.92	0.92	0.96
2	3.10	2.48	1.91
3	4.06	3.39	2.99
4	4.98	4.24	3.97
5	6.01	5.06	4.93

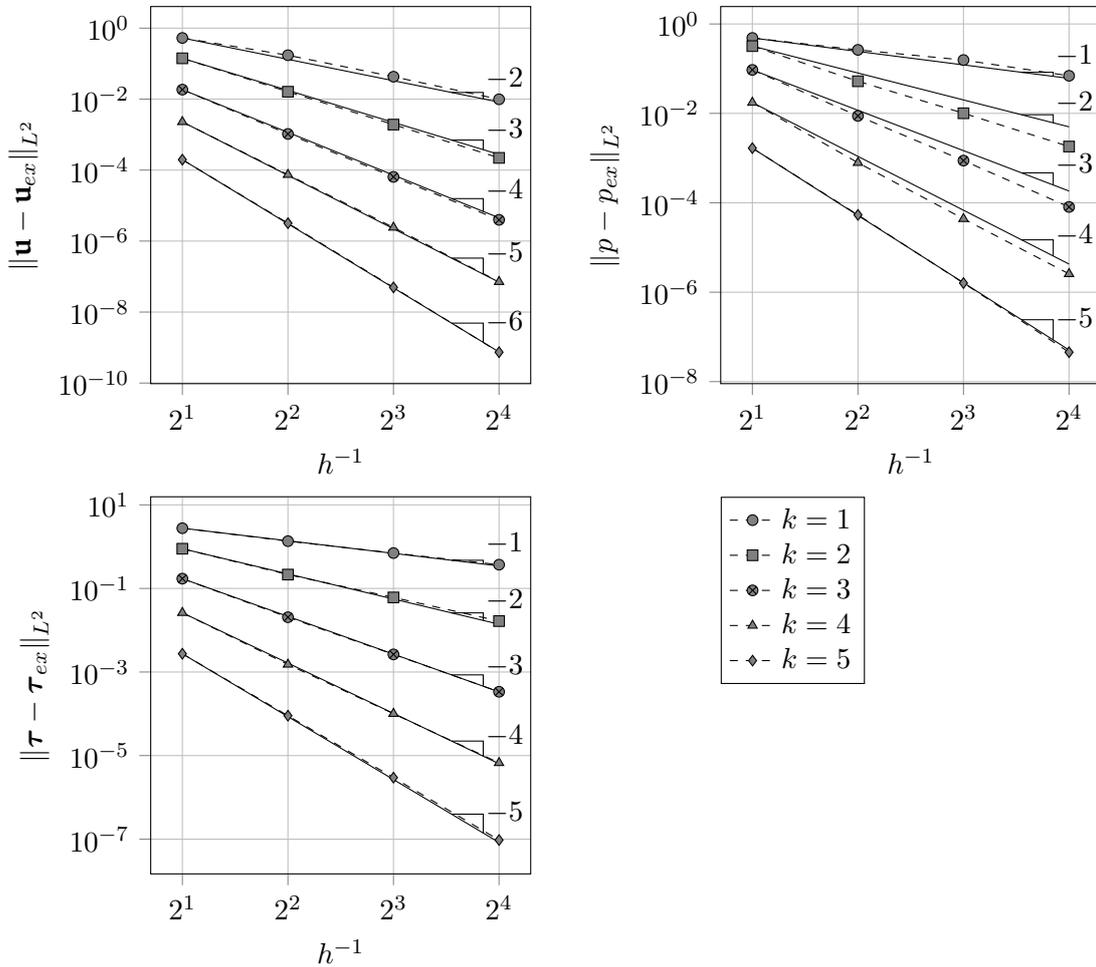


Figure 7.1.: h^k -convergence for the L_2 -error against the analytical solution in Eq. (7.1) up to a polynomial order of $k = 5$ for the velocity \mathbf{u} and the stresses $\boldsymbol{\tau}$ and of $k = 4$ for the pressure p . The solid lines mark the expected order of convergence of $k + 1$ for \mathbf{u} and p and k for $\boldsymbol{\tau}$.

because problems arise in resolving steep gradients in velocity and stress and very thin boundary layers as well as a very fine nearly singular beam in the wake of the cylinder (Claus and Phillips, 2013). Without stabilizing techniques most numerical approaches diverge for Weissenberg numbers greater than 0.7 (e.g. Kim et al., 2004). The lack of convergence is due to either numerical errors propagating into the wake of the cylinder or physical instabilities inherent in the viscoelastic model which lead to an exponential increase of the stresses at the rear stagnation point behind the cylinder (Claus and Phillips, 2013).

In newer times, the physical instabilities in the viscoelastic flow around the cylinder have come into focus of research. Experimentally, such instabilities were confirmed and analysed in detail earlier (McKinley et al., 1993). There seems to be an unstable detaching boundary layer on the cusp of the cylinder leading to a transitional flow which is unsteady beyond $Wi = 1$ and three-dimensional structures arise. This means that with increasing Weissenberg number a convergent numerical solution cannot be found, especially, when fine meshes and high-order accurate methods try to capture the evolving structures close to the cylinder. This could be the reason, why solutions on coarser meshes might have less difficulty to converge (Claus and Phillips, 2013).

Here, we present some results for the steady and unsteady viscoelastic flow for the confined cylinder benchmark problem which are extracted from the authors publication currently under review (Kikker et al., 2020). If there are problems finding a convergent solution, we can use the cell indicator combined with artificial diffusion (Persson and Peraire, 2006). The findings are compared to results obtained from other numerical methods.

The confined cylinder benchmark problem is a two-dimensional numerical simulation of viscoelastic flow around a cylinder (radius $r = 1$) immersed in a narrow channel (height $\bar{h} = 4$) with a blocking ratio of $\frac{r}{\bar{h}} = \frac{1}{2}$ (Fig.7.2). The inflow Dirichlet BC for the velocity is:

$$u_x = \frac{3}{2} \left(1 - \frac{y^2}{4} \right), \quad u_y = 0, \quad (7.2)$$

such that the bulk velocity u_b is 1 at the inflow. We have no Dirichlet values for the stresses since this is a local field and in all boundary fluxes we only use the inner value τ^- resulting from the velocity field. Since we expect the flow to be symmetric for the steady-state case, we consider only half of the channel with a free-slip BC at the centre-line in order to save computational time. At the walls of the channel and on the cylinder surface we have a no-slip BC with an impermeable wall. The outflow is a pressure outlet, with the pressure and the velocity gradients set to zero.

We use a body-fitted curved elements grid with polynomial order five, which is non-equidistant with gradients towards the cylinder (Fig. 7.3). In case of curved elements grids the polynomial order of the polygons in the grid needs to be higher than the computational polynomial order (Bassi and Rebay, 1997). For the convergence study we have different refinement levels starting from 16 elements dividing the channel height at the inlet up to 128 elements for the finest grid. The amount of nodes of the coarsest grid is doubled for each refinement level. Only for the parts in x -direction of the channel we choose a refinement factor of 1.5. An overview over the different meshes can be seen in Tab. 7.3. The DOF are listed in Tab. 7.4. Increasing DOF, especially for finer meshes and high polynomial orders, cause memory issues at computation, since in this implementation the system is fully coupled. This is particularly the case for the direct solvers when decomposing the comparably dense matrix of the non-saddle-point problem. To solve this in future, iterative linear solvers for High Performance Computing (HPC)

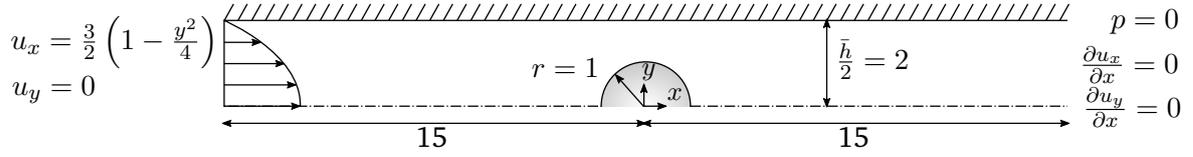


Figure 7.2.: Computational domain with BC for velocities.

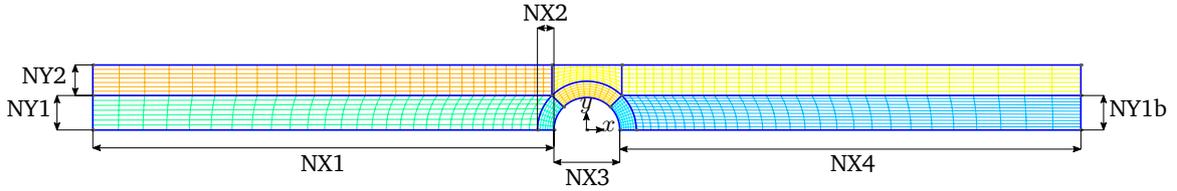


Figure 7.3.: Mesh mesh_1 with location of number of nodes as referred to in table 7.3.

are currently under development within the BoSSS framework.
The dimensionless parameters are defined as follows:

$$\text{Re} = \frac{\rho u_b r}{\eta_0}, \quad \text{Wi} = \lambda_1 \frac{u_b}{r}, \quad \beta = \frac{\eta_s}{\eta_0}. \quad (7.3)$$

They are chosen, if not stated differently, as follows: $\beta = 0.59$, $\text{Re} = 0, 0.01, 0.1, 1$, $\text{Wi} = 0, \dots, 1$. Apart from the convergence study (Sec. 7.2.1) all calculations were made with the comparably coarse mesh_1 and a polynomial degree for the velocity and stresses of $k = 4$. For all calculations apart from Sec. 7.2.4 we have $\text{Re} = 0$. For all steady calculations we use an implicit Euler scheme with one pseudo-time-step of $\Delta t = 10^6$, for the unsteady calculations in Sec. 7.2.3 we use the BDF2 scheme (Eq. 4.48) with a time-step size of $\Delta t = 10^{-2}$.

7.2.1. Convergence Study

We examine our solver in the case of the confined cylinder problem for convergence against the solution of the finest grid for different polynomial degrees in the approximation spaces ($k = 1 \dots 4$) for $\text{Re} = 0$. For $\text{Wi} = 0$ and thus, a Newtonian fluid flow and $\text{Wi} = 0.2$ the results

Table 7.3.: Amount of nodes of the computational meshes in different regions for the convergence study and corresponding refinement factor for that direction.

mesh	NX1	NX2	NX3	NX4	NY1	NY1b	NY2
mesh_0	20	4	6	20	4	6	4
mesh_1	30	8	12	30	8	12	8
mesh_2	45	16	24	45	16	24	16
mesh_3	68	32	48	68	32	48	32
factor	1.5	2	2	1.5	2	2	2

Table 7.4.: Number of cells and of degrees of freedom for different mesh sizes and polynomial degrees with k for the velocity and stresses and $k - 1$ for the pressure.

mesh	no of cells	$k = 1$	$k = 2$	$k = 3$	$k = 4$
mesh_0	320	5.120	10.560	17.920	27.200
mesh_1	1.208	19.328	39.864	67.648	102.680
mesh_2	4.252	68.032	140.316	238.112	361.420
mesh_3	14.712	235.392	485.496	823.872	1.250.520

Table 7.5.: Polynomial degree of the discretization and experimental order of convergence (EOC) for both Weissenberg numbers and different dependent variables.

k	EOC _{\mathbf{u}}		EOC _{p}		EOC _{$\boldsymbol{\tau}$}	
	Wi = 0	Wi = 0.2	Wi = 0	Wi = 0.2	Wi = 0	Wi = 0.2
1	2.17	2.18	1.76	1.75	1.11	1.33
2	3.46	2.65	2.03	2.24	2.02	1.92
3	4.12	3.70	2.76	2.79	2.92	2.70
4	5.27	3.77	4.08	3.00	4.03	3.10

are shown in Fig. 7.4 and the convergence rates are listed in Tab. 7.5. For the Newtonian case, all convergence rates for the L_2 -Norm are equal or greater than $k + 1$ for the velocity and pressure or k for the stress, as is expected for the LDG scheme used for discretization (Cockburn et al., 2002). For the viscoelastic case there are small deviations in the L_2 -error for the finer grids and higher polynomial degrees such that the convergence rates of $k + 1$ and k cannot be fulfilled completely for all dependent variables.

7.2.2. Results for Steady Flow Simulation

Cylinder drag

As a measure to compare the accuracy of the method with other methods from literature, the dimensionless drag force of the cylinder in the flow for different Weissenberg numbers is used:

$$F := \int_{\Gamma} \mathbf{T} \cdot \mathbf{n} \, dS, \quad (7.4)$$

where F is the dimensionless drag force calculated as the surface integral of the total stress tensor

$$\mathbf{T} = -p\mathbf{I} + \frac{\beta}{\text{Re}} \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T \right) + \frac{1}{\text{Re}} \boldsymbol{\tau}. \quad (7.5)$$

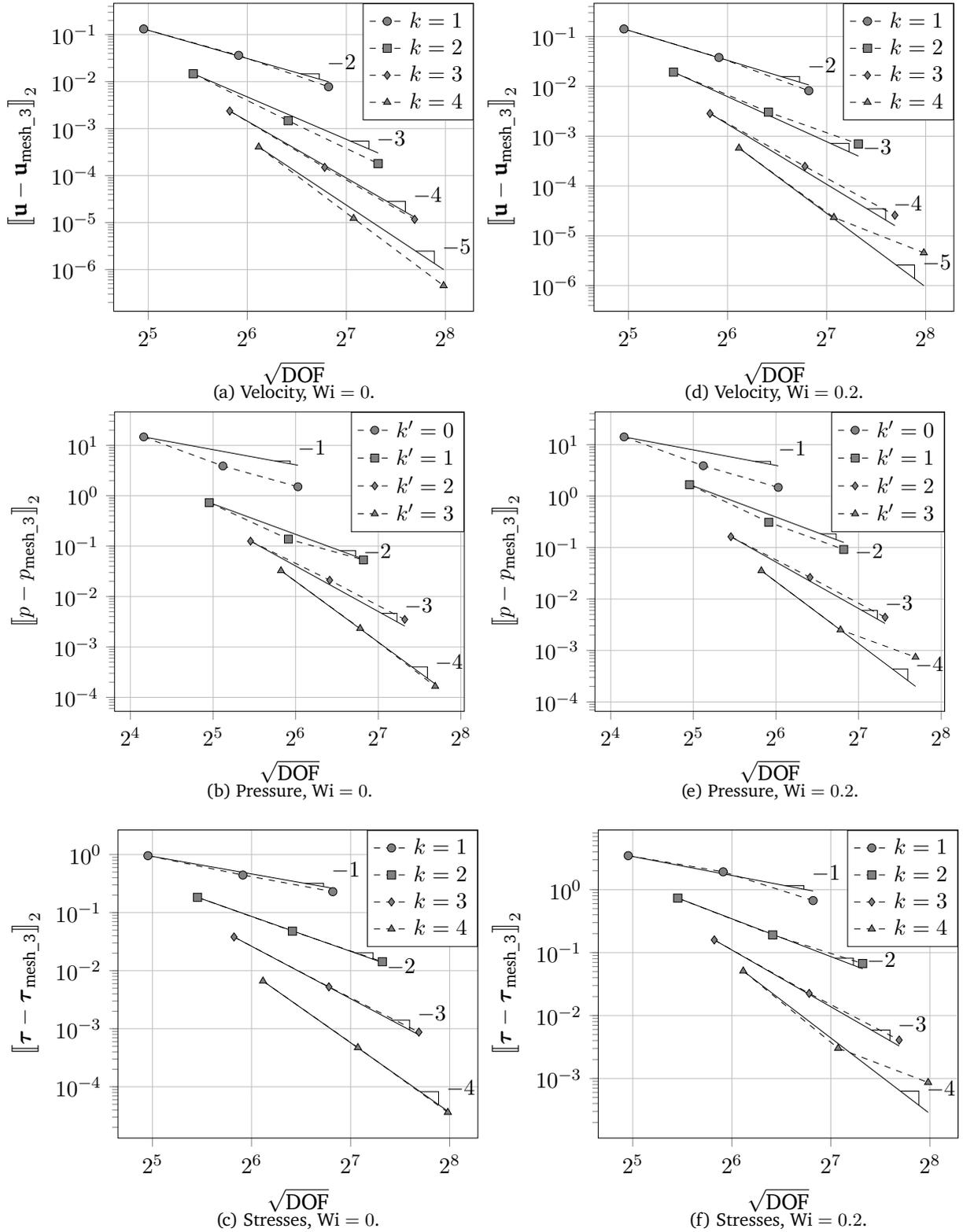


Figure 7.4.: Convergence study for $Wi = 0$ (left) and $Wi = 0.2$ (right) in the L_2 -Norm compared to the DOF of the finest mesh. The solid lines show the expected convergence rates with $k + 1$ for velocity and pressure and k for the stresses.

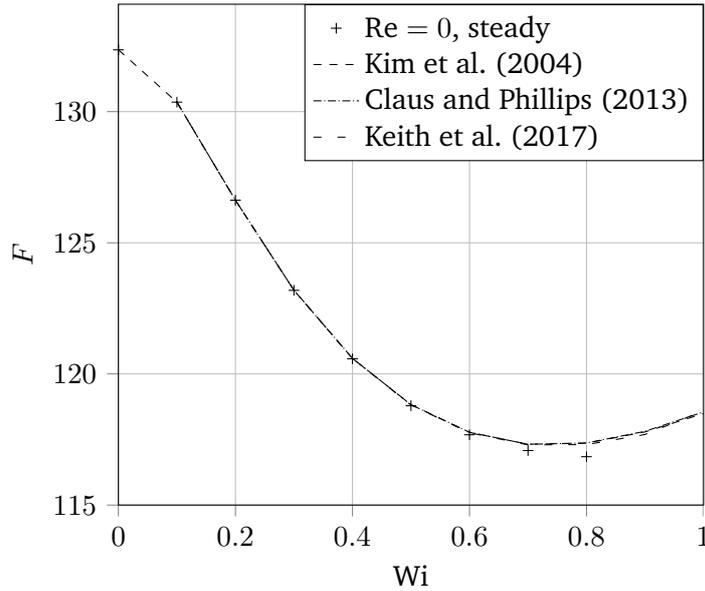


Figure 7.5.: Dimensionless drag force of the confined cylinder for different Weissenberg numbers for steady calculations (+) compared with unsteady calculations from exemplary literature (Kim et al., 2004; Keith et al., 2017; Claus and Phillips, 2013). For $Wi \leq 0.6$ we have good agreement. Afterwards, unsteady effects cause errors in the unstable steady state solution as expected.

The resulting drag forces for different Weissenberg numbers are plotted in Fig. 7.5, listed in Tab. 7.6 and compared to selected literature (Kim et al., 2004; Keith et al., 2017; Claus and Phillips, 2013). For smaller Weissenberg numbers ($Wi \leq 0.6$) we are in good agreement with the values obtained in other studies. Our drag force values are lower for the higher Weissenberg numbers due to lower absolute values for the normal stresses along the cylinder surface where the drag is calculated (Fig. 7.7). This is due to the fact that our steady simulations naturally do not capture the unsteady behaviour which is observed for $Wi > 0.6$. For the same reason we could not conduct a steady convergent solution for $Wi > 0.8$.

Flow Behavior

In Fig. 7.6 elevate plots for the velocity and the stresses for the whole domain are shown. The velocity u_y is accelerated in the narrowing between the channel wall and the cylinder and we have a small vertical velocity u_y due to the displacement of the fluid along the cylinder. At the cusp of the cylinder and at the channel wall, we can see high stress peaks, especially in the normal stress τ_{xx} , and a second peak in the wake of the cylinder. The values of the normal stress τ_{xx} are magnitudes greater than the values of τ_{yy} .

The flow behaviour of the stress component τ_{xx} along the symmetry line and on the cylinder surface is further investigated. As can be seen in Fig. 7.7, our results are in very good agreement with the work of Claus and Phillips (2013) concerning the wake of the cylinder. Along the cylinder surface the resulting normal stress τ_{xx} is for higher Weissenberg numbers not as high as in the work of Claus and Phillips (2013), which can be due to the fact that we

Table 7.6.: Dimensionless drag force of the confined cylinder for different Weissenberg numbers for steady calculations compared with unsteady calculations from exemplary literature [a] Kim et al. (2004), [b] Claus and Phillips (2013), and [c] Keith et al. (2017).

Wi	F	[a]	[b]	[c]
0	132.357	132.36	-	-
0.1	130.363	130.36	130.364	130.3618
0.2	126.625	126.62	126.626	126.6241
0.3	123.188	123.19	123.192	123.1897
0.4	120.577	120.59	120.593	120.5885
0.5	118.789	118.83	118.826	118.8132
0.6	117.680	117.77	117.776	117.7581
0.7	117.079	117.32	117.316	117.2951
0.8	116.844	117.36	117.368	117.3057
0.9	-	117.79	117.812	117.6907
1.0	-	118.49	118.550	118.5970

are comparing steady solutions with steady-state solutions from an unsteady computation from literature. The stress component τ_{xx} increases along the cylinder surface up to the cusp of the cylinder and then decreases to zero at the rear stagnation point. In the following a second steep gradient can be seen, forming a tail in the wake of the cylinder with a peak directly behind the rear stagnation point. The two peaks grow for increasing Weissenberg numbers.

Dou and Phan-Thien (2007) developed an inflection point theory for a detaching boundary layer at the cusp of the cylinder for increasing $Wi \geq 0.7$. Claus and Phillips (2013) found many indications supporting that theory. We investigated our results obtained from steady calculations for these indications and are in good agreement.

Dou and Phan-Thien explained the appearance of an inflection point in the velocity profile in the boundary layer at the cusp of the cylinder with a non-constant and increasing pressure in the boundary layer for increasing Weissenberg number. For $Wi > 0.6$ the resulting pressure gradient from within the shear layer to outside $\left(\frac{\partial p}{\partial y}\right)$ is large and causes a maximum in the velocity gradient with a resulting inflection point in the velocity (Dou and Phan-Thien, 2007). This behavior can be measured by the ratio between the stream-wise and normal energy gradients, which are proportional to the ratio of the corresponding pressure gradients for negligible kinetic energy. Hence, the angle between cross-stream and stream-wise pressure is a measure for the probability for an inflection point in the velocity (Claus and Phillips, 2013). In Fig. 7.8 the pressure distribution at the cusp of the cylinder is shown for different Weissenberg numbers. The angle described above increases for increasing Weissenberg numbers, which is in good agreement with Claus and Phillips (2013).

Further indication for the appearance of a velocity inflection is given by the behavior of the velocity itself at the cusp of the cylinder. The distribution of the velocity u_y increases for increasing Weissenberg number close to the cusp of the cylinder and decreases in the upper

part close to the channel wall (Fig 7.9). At $y \approx 1.01$ there is a kink in the slope of the velocity when leaving the boundary layer. This kink is not at a discontinuous element edge but within a cell and was also detected by Dou and Phan-Thien (2007) and Claus and Phillips (2013), but at $y \approx 1.02$, so that it is possibly no numerical artefact. This change in slope increases for increasing Weissenberg numbers. Dou and Phan-Thien assume that it causes fluid elements to leave the boundary layer and therefore, disturb the flow. This kind of disturbance can be transported downstream and amplified, leading to a transition to unsteady behaviour behind the cylinder.

Considering the distribution of the velocity u (Fig. 7.10) it can be seen that the maximum velocity increases with increasing Weissenberg numbers. Furthermore, the velocity is decreased close to the cylinder surface with increasing Weissenberg number.

7.2.3. Results for Unsteady Flow Simulation

For $Wi \geq 0.5$ we performed unsteady simulations with a BDF2 scheme and $\Delta t = 10^{-2}$. Whereas for $Wi = 0.5$ a stable steady state solution can be achieved over a long time period, for higher Weissenberg numbers the simulation stops converging at a certain time-step without diverging (Fig. 7.11). For $Wi = 0.6$ this can be observed at time-step 1586 whereas for $Wi = 1.0$ this time-step is already time-step 864.

As can be seen in detail in Fig. 7.11, an apparent steady-state solution is reached for the lower Weissenberg numbers up to $Wi = 0.7$ before the solution stops converging, but for the higher Weissenberg numbers the drag coefficient starts to oscillate immediately.

Further investigation about the lack of convergence for unsteady calculation with higher Weissenberg numbers is needed.

7.2.4. Results for Different Reynolds Numbers

In this section we show the behaviour of the steady solution for increasing Reynolds number and thus, examine the influence of inertia on the flow. As we can see in Fig. 7.12, the drag of the cylinder increases for higher Reynolds numbers and the drag reduction effect caused by the viscoelasticity is reduced. This is due to the increased velocity gradients caused by the inertia of the fluid. For lower Weissenberg numbers our steady results are in good agreement with the literature (Tab. 7.7; Kim et al., 2004; Claus and Phillips, 2013; Keith et al., 2017). All further studies were made for $Wi = 0.6$. The velocity u_y increases for increasing Reynolds number close to the cusp of the cylinder and decreases with increasing Reynolds number close to the channel wall. This behavior is amplified for viscoelastic flow as showed in Fig. 7.13. Here, the effect is shown for a Newtonian fluid as well as the Oldroyd B fluid with $Wi = 0.6$. The normal stress τ_{xx} in Fig. 7.14 decreases only slightly with increasing Reynolds number at the cusp of the cylinder, but in the wake of the cylinder a large decrease is notable for $Re = 1$.

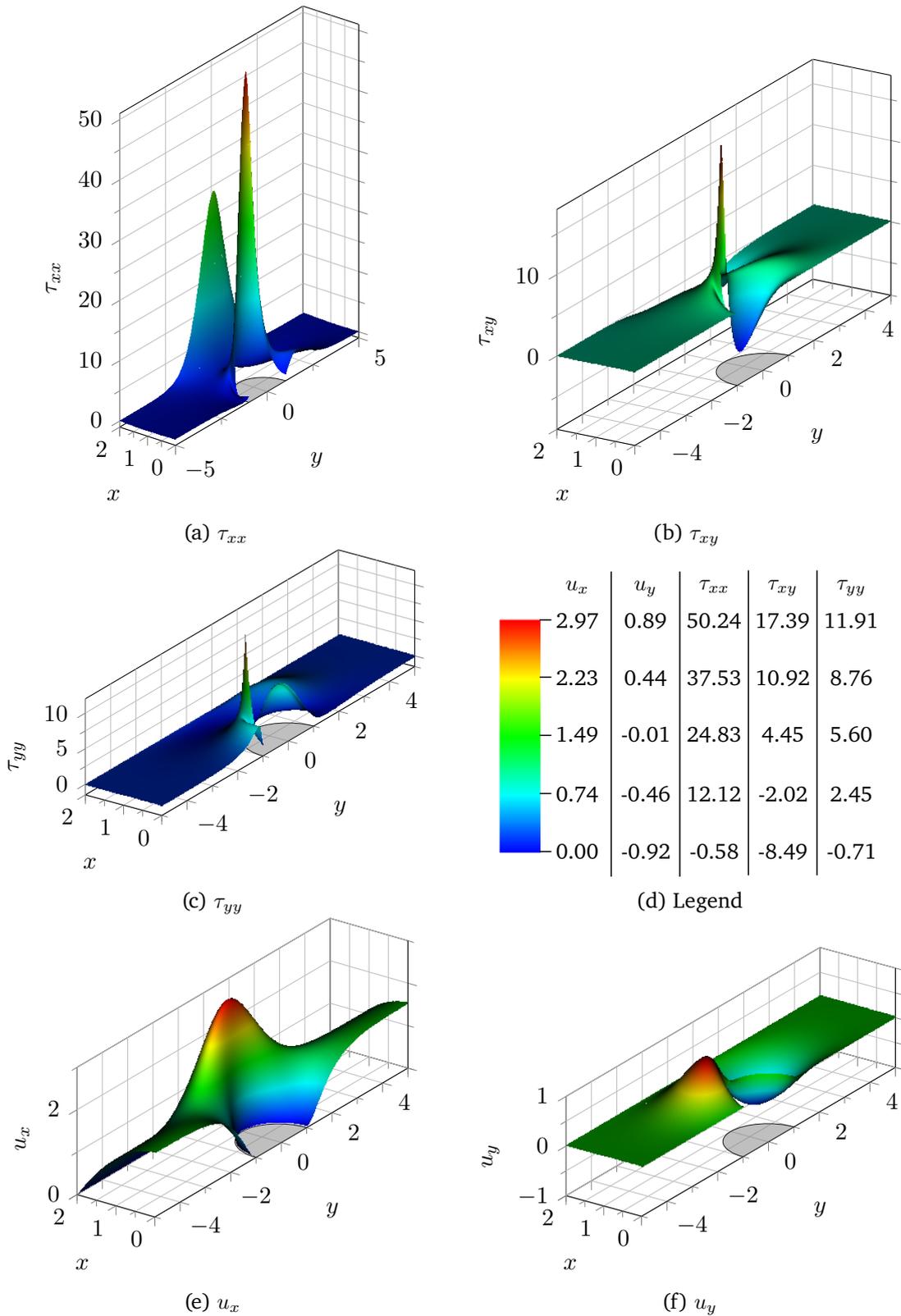


Figure 7.6.: Elevate plots of the stress and velocity profiles over the domain $-5 \leq x \leq 5$ for $Wi = 0.3$. The acceleration of the fluid and the high stress peaks in the narrow at the cusp of the cylinder as well as the stress peak in the wake of the cylinder are clearly visible.

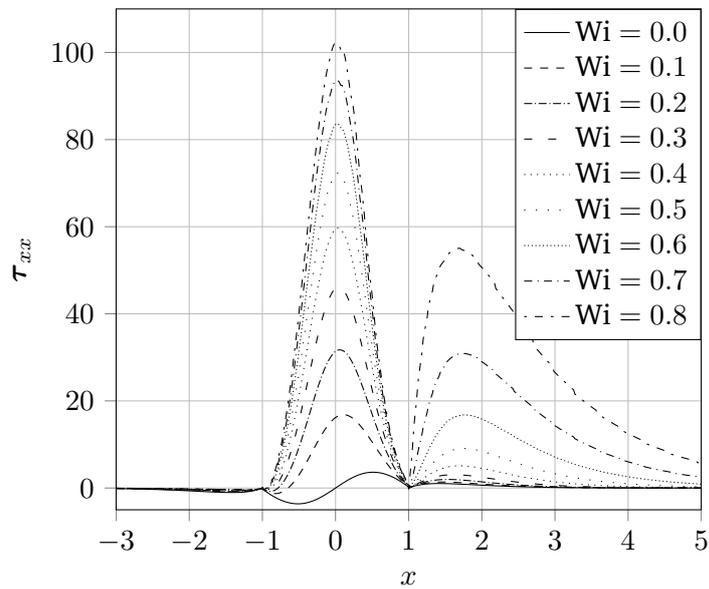


Figure 7.7.: Normal stress τ_{xx} at the symmetry line and on the cylinder surface in the interval $[-1...1]$. For larger $Wi > 0.8$ no steady solution could be accomplished.

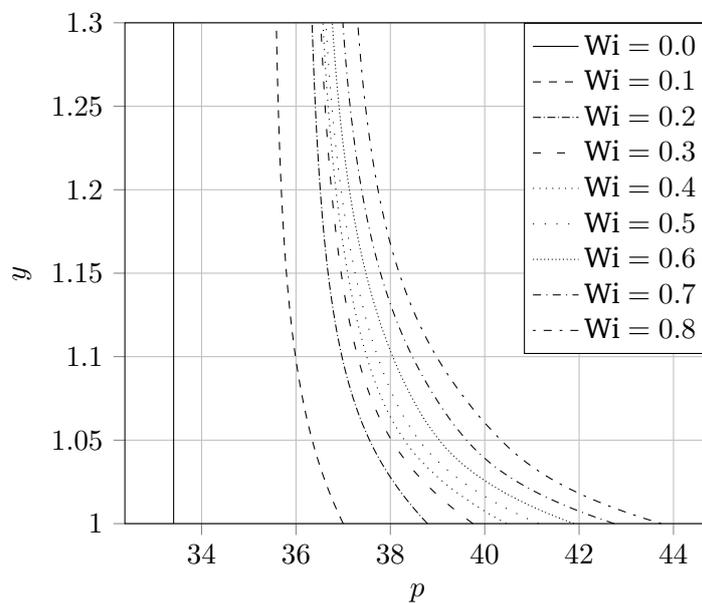


Figure 7.8.: Pressure distribution on the cusp of the cylinder. For $Wi > 0.8$ no convergent steady solution could be accomplished. It can be seen that the angle between the cross-stream and the stream-wise pressure derivative becomes larger for higher Weissenberg numbers.

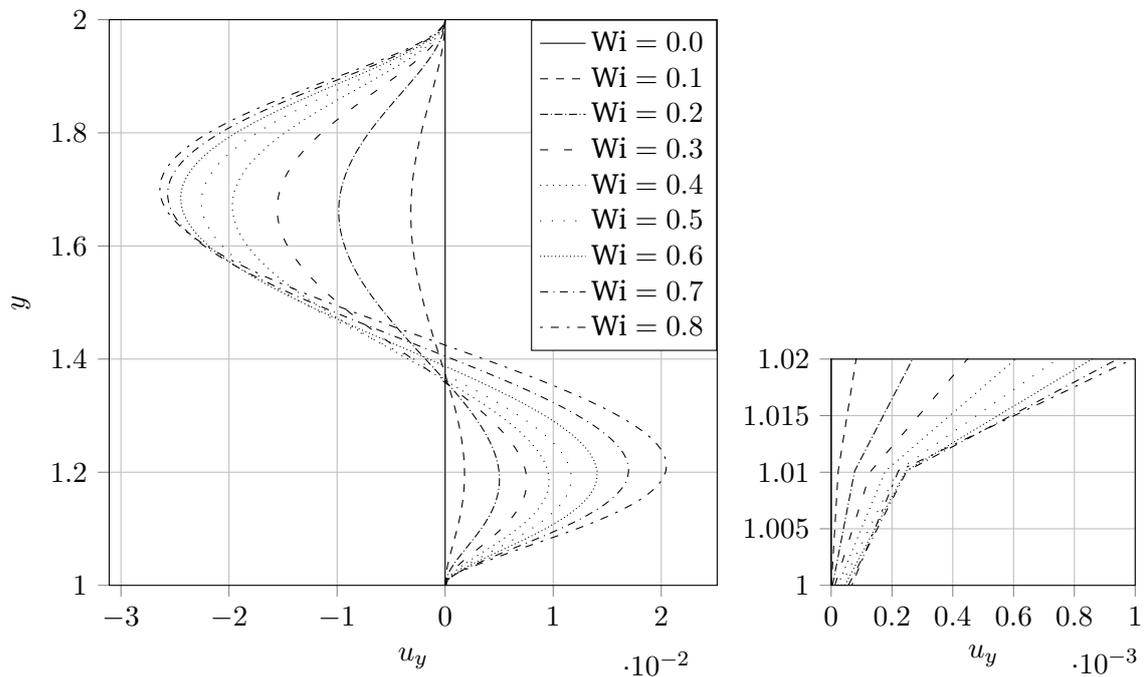


Figure 7.9.: Distribution of the velocity u_y on the cusp of the cylinder. For larger $Wi > 0.8$ no convergent steady solution could be accomplished. It can be seen in the detail plot that there is a kink in the distribution close to the cylinder wall.

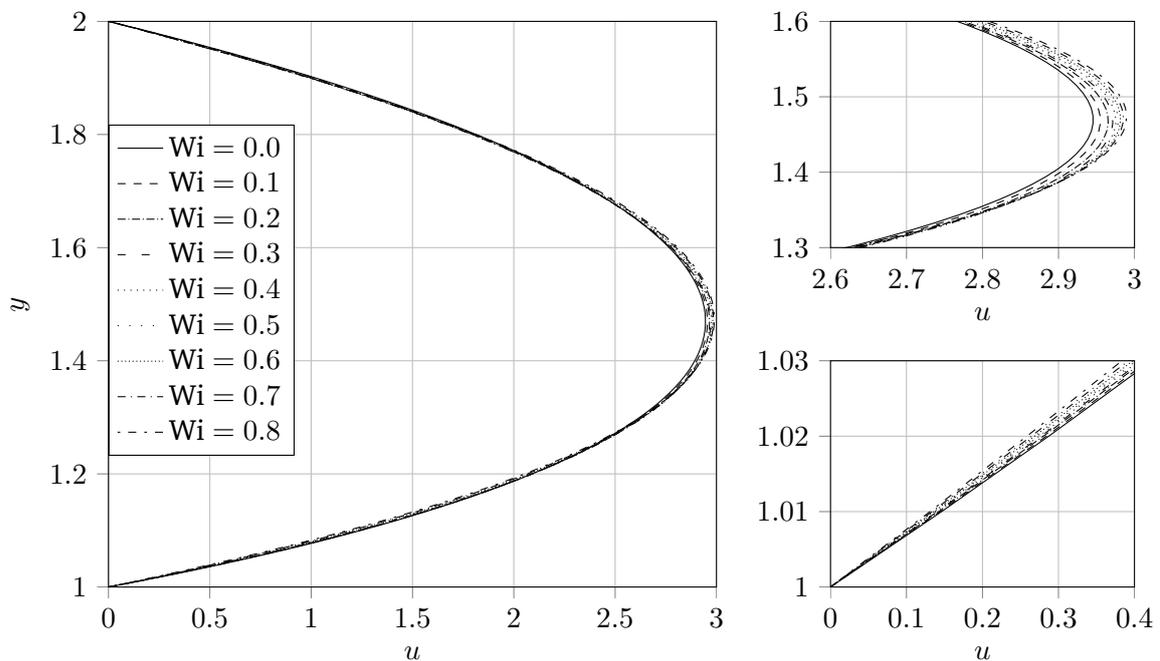


Figure 7.10.: Distribution of the velocity u on the cusp of the cylinder. For larger $Wi > 0.8$ no convergent steady solution could be accomplished. On the detail plots it can be seen that the maximum velocity increases for increasing Weissenberg numbers and the velocity decreases at the cusp of the cylinder.

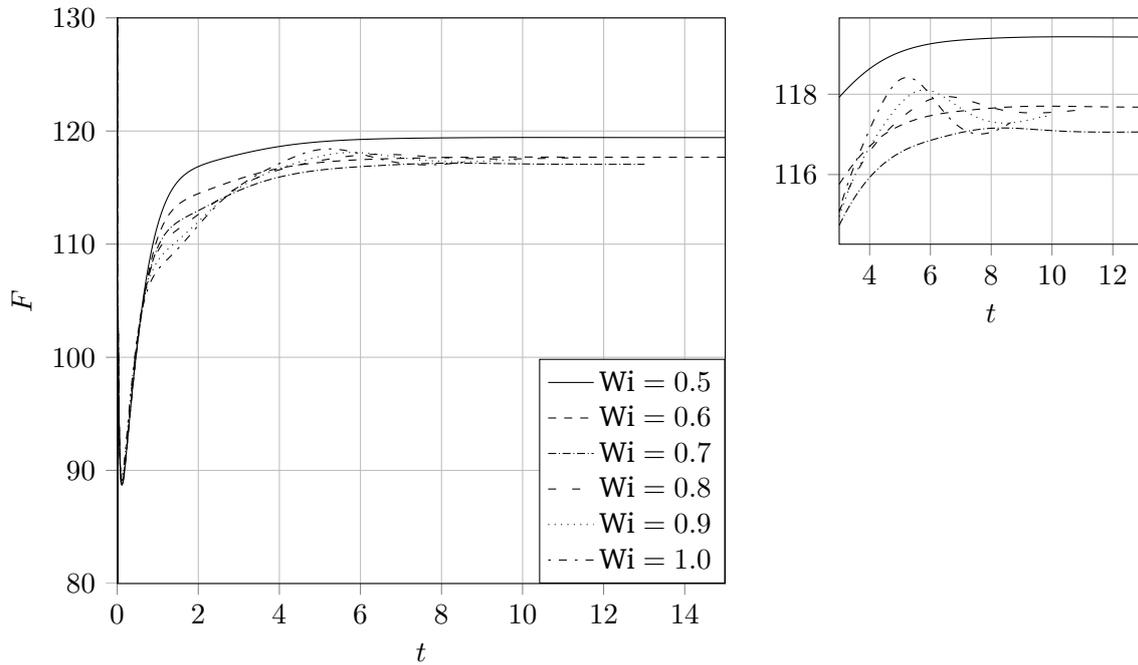


Figure 7.11.: Evolution in time of the drag force of the confined cylinder for different Weissenberg numbers for unsteady calculations. For higher $Wi \geq 0.6$ there is in the detail plot an oscillatory behaviour which might be caused by unsteady effects due to velocity inflection in the boundary layer.

Table 7.7.: Dimensionless drag force of the confined cylinder for different Weissenberg numbers for steady calculations for different Reynolds numbers compared with unsteady calculations from exemplary literature [a] Claus and Phillips (2013), and [b] Keith et al. (2017).

Wi	$Re = 0.01$	[a]	[b]	$Re = 0.1$	[a]	[b]	$Re = 1$	[a]	[b]
0.1	130.363	130.364	130.363	130.367	130.368	130.367	130.607	130.609	130.608
0.2	126.626	126.627	126.626	126.635	126.636	126.635	126.936	126.938	126.937
0.3	123.189	123.194	123.192	123.206	123.211	123.210	123.593	123.597	123.596
0.4	120.579	120.595	120.593	120.606	120.622	120.620	121.093	121.106	121.106
0.5	118.792	118.831	118.827	118.830	118.868	118.865	119.423	119.460	119.457
0.6	117.685	117.781	117.775	117.734	117.831	117.823	118.424	118.542	118.538
0.7	117.085	117.323	117.308	117.143	117.387	117.372	117.918	118.233	118.222
0.8	116.851	117.379	117.277	116.920	117.459	117.373	-	118.455	118.397
0.9	116.878	117.827	117.559	-	117.925	117.684	-	119.096	118.874
1.0	-	118.563	118.130	-	118.697	118.224	-	120.057	120.146

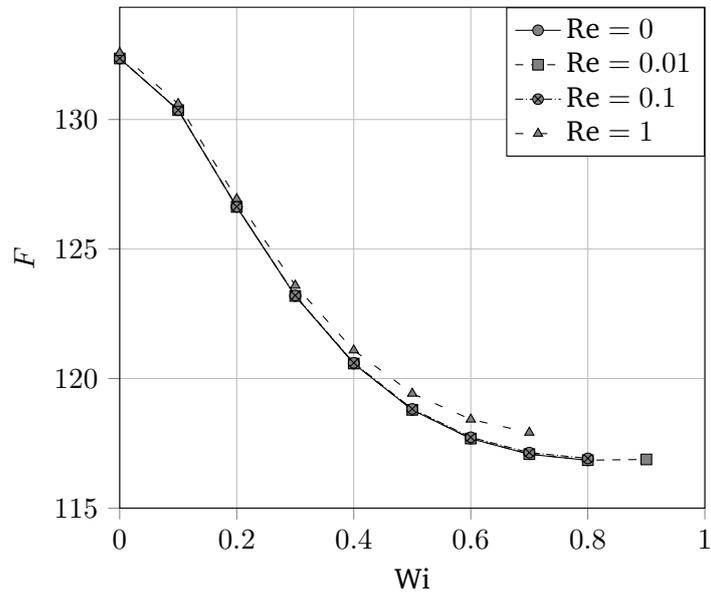


Figure 7.12.: Drag coefficient depending on the Weissenberg number for different Reynolds numbers.

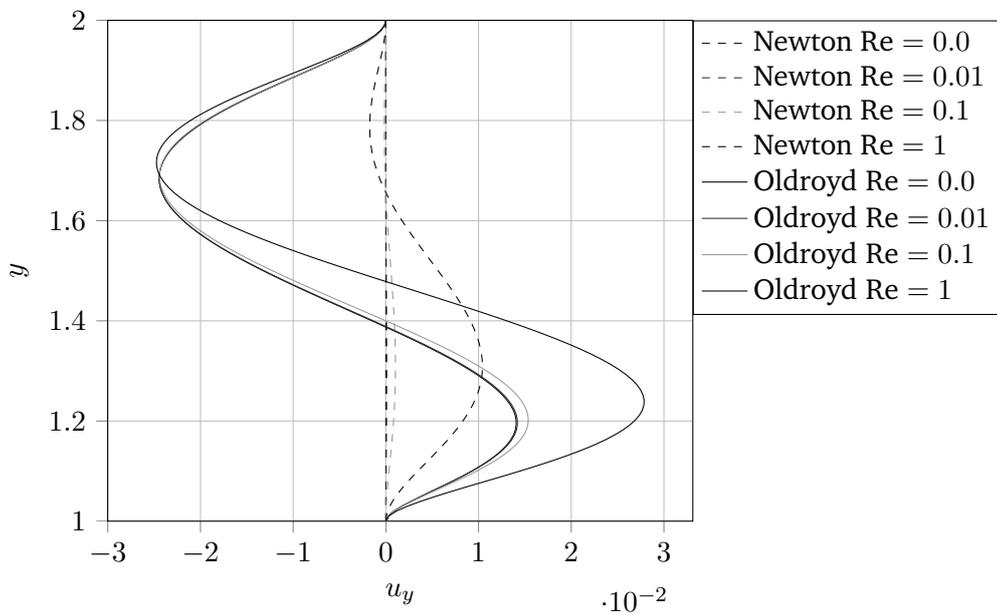


Figure 7.13.: Distribution of the velocity u_y on the cusp of the cylinder for $Wi = 0.6$ and Newtonian flow for different Reynolds numbers.

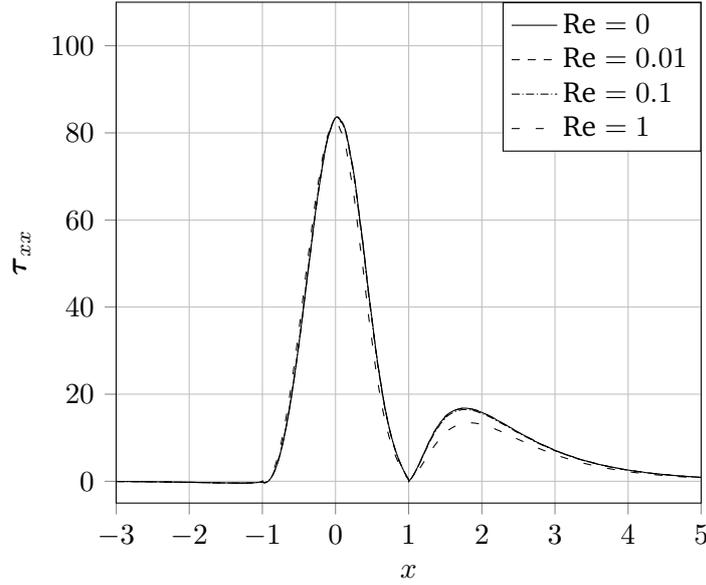


Figure 7.14.: Normal stress τ_{xx} at the symmetry line and on the cylinder surface in the interval $[-1\dots 1]$ for $Wi = 0.6$ for different Reynolds numbers.

7.3. Droplet in Shear Flow

In order to validate the multi-phase XDG solver, the work of Chinyoka et al. (2005) is taken into account. They examine the behaviour of a droplet in a shear flow conducted by a channel with moving walls in a two-dimensional setting. In different simulations the drop deformation and the drop shape is investigated for a Newtonian droplet in a Newtonian fluid (NN), a Newtonian droplet in a viscoelastic fluid (NV), and a viscoelastic droplet in a Newtonian fluid (VN). Here, the inside of the droplet is always referred to as phase \mathfrak{B} and the fluid outside of the droplet as phase \mathfrak{A} . The dimensions of the channel and the used BC are presented in Fig. 7.15. The channel height is 1 such that the initial radius of the droplet is $r = \frac{\bar{h}}{8} = 0.125$ and the channel length is $l = 16r = 2$. The channel top wall is moving with $u_T = 1$ in x -direction and the bottom wall is moving with $u_B = -u_T$ and we have a zero pressure inlet and outlet. The channel dimensions are chosen large enough compared to the droplet radius such that the droplet behaves like being immersed in an infinite medium with a constant shear rate $\dot{\gamma} = \frac{u_T - u_B}{h}$ (Chinyoka et al., 2005). For the discretization the domain is divided into 16 equidistant elements in y -direction and into 32 cells in x -direction. We use the curvature refined adaptive mesh refinement with 3 refinement levels (Fig. 7.16). In the dimensionless setting, the following parameters are varied and compared: the Reynolds number (Eq. 2.65), the Deborah number (Eq. (2.74)), a viscosity ratio $m = \frac{\beta_{\mathfrak{A}}}{\beta_{\mathfrak{B}}}$ and the capillary number:

$$Ca := \frac{r\dot{\gamma}\beta_{\mathfrak{B}}}{\sigma}, \quad (7.6)$$

which relates the surface tension σ or the Weber number for an isotropic surface stress tensor with constant surface tension to the problem. In this case $\sigma = \frac{1}{We}$. The Deborah number is defined in Chinyoka et al. (2005) using the shear rate and thus, is equivalent to our definition

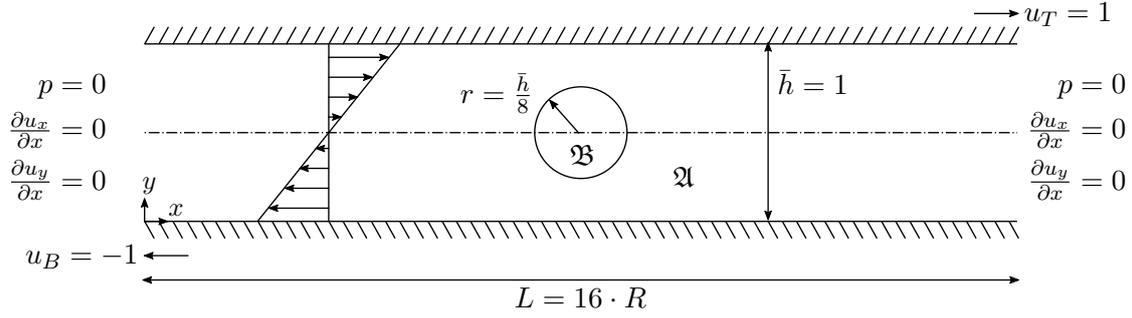


Figure 7.15.: Computational domain of a droplet in shear flow with moving walls. The measurements and BC are filled in.

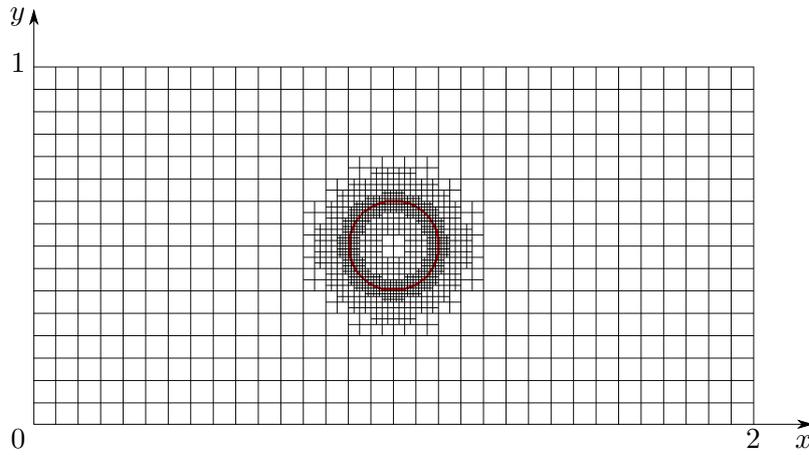


Figure 7.16.: Mesh of the droplet in shear flow after initial refinement with three refinement levels depending on the curvature of the interface. The interface represented by the zero level-set ($\varphi = 0$) is marked in red.

for the Weissenberg number (Eq. 2.65).

In this first feasibility study, we use the Boussinesq-Scriven model for the surface stress and choose $\tilde{\Lambda}_\gamma = \eta_0 \cdot 2 \cdot \sigma$ and $\eta_\gamma = \eta_0 \cdot \sigma$ whereas η_0 is derived from the Reynolds number. We consider a flow with $\text{Ca} = 60$, $\text{Re} = 0.3$, $m = 1$ and $\eta_s = \eta_p$ which means $\beta = 0.5$. We define for all viscoelastic phases a moderate Weissenberg number of 0.3. The polynomial order for discretization is $k = 2$ for velocity and stresses and $k' = 1$ for the pressure for the low-order simulation, and $k = 4$ for velocity and stresses and $k' = 3$ for the pressure for the high-order simulation, respectively. We have a time-step size of $\Delta t = 10^{-3}$ and choose the splitting method for the time integration. The high-order simulations have computational issues in the single-core computation and therefore, already stop at time $t = 0.021$.

Deformation behaviour of the droplet

We are interested in the behaviour of the shearing droplet and into the question, whether and when the droplet reaches a steady state while shearing. Therefore, we consider the

deformation of the droplet measured by the deformation parameter D with:

$$D := \frac{r_{\max} - r_{\min}}{r_{\max} + r_{\min}} \quad (7.7)$$

where r_{\max} and r_{\min} are the longest and shortest distances of the interface to the drop center, respectively. In Fig. 7.17 the deformation can be seen for the (NN)-, the (VN)-, and the (NV)-case for the low-order simulation. The calculations for all three test cases stop in a very early state because the agglomeration of the cut-cells fails due to discontinuities in the zero level-set. Such discontinuities of the zero level-set, which are small in amplitude, can also be seen in Fig. 7.17. They are due to the zero level-set crossing cell boundaries or a regionally limited large deformation of the drop. The simulations stop at time $t = 0.089$ (NN), $t = 0.081$ (NV), and $t = 0.051$ (VN), respectively. It is recognizable, that the large stresses and therefore, the large stress differences between the inside and the outside of the droplet compared to simulations of Newtonian flow solely using SIP, lead to numerical artefacts at the interface prior to a fail of the agglomeration. This also means that we do not achieve a steady state which is reached at $t \approx 3.00$. The results which are compared in Chinyoka et al. (2005) are at the time $t = 6.00$. Further investigation is needed to enhance the stability of the level-set algorithm against high shear rates and large stress differences.

In Fig. 7.18 the zero level set contours are plotted for all three test cases for time $t = 0.040$ and for the (NN)- and the (VN)-simulation at time $t = 0.075$. It can be seen, as already shown in Fig. 7.17, that for the (VN)-simulation the deformation is the highest and for the (NV)-simulation the lowest, although differences are still small. Compared to the (NN)-simulation, the droplet of the (NV)-simulation is stretched more and the droplet of the (VN)-simulation less. At time $t = 0.075$, kinks in the zero level-set contour are already noticeable for both test cases leading to the previously described agglomeration failure.

The increased stretching of the Newtonian droplet in the viscoelastic fluid in this early stage of the simulation can be explained by the velocity vector distribution in Fig. 7.19. These plots are showing the pseudocolouring of the shear stress τ_{xy} and the velocity vectors for the high-order (NV)- and (VN)-simulation at time $t = 0.020$. It can be seen, that for the (NV)-case the velocity of the fluid reaching the droplet is pointing in the direction of the droplet, increasing the pressure in this region, whereas for the (VN)-case the velocity vectors are pointing outward.

The ability of simulating droplets with viscoelastic fluids could be shown with this test case. The early stage of the simulations does not allow to show major viscoelastic effects on the droplet or compare the results with the model study of Chinyoka et al. (2005). Furthermore, no steady-state solution was reached. However, some insights in the significantly different behaviour of two-phase flows including viscoelasticity could be given. The outlined problems concerning the level-set are also occurring to a lesser extent when simulating Newtonian droplets solely using SIP. Different approaches for solving these issues are currently part of the scientific work conducted in the BoSSS group.

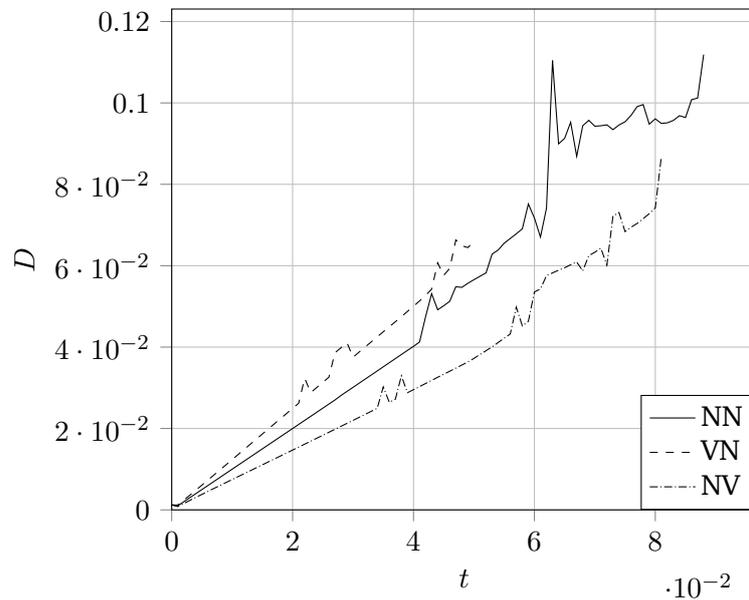


Figure 7.17.: Drag coefficient depending on the Weissenberg number for different Reynolds numbers.

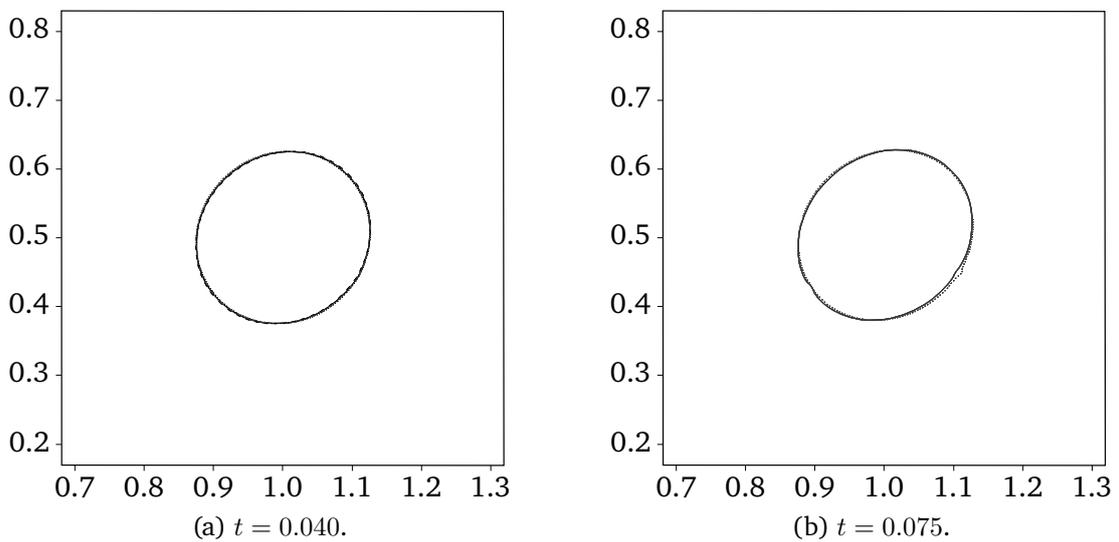


Figure 7.18.: Contour of the zero level-set at different times of the (NN)-simulation (solid line), the (NV)-simulation (dashed), and the (VN)-simulation (dotted). At $t = 0.075$ the (NV)-simulation had already failed.

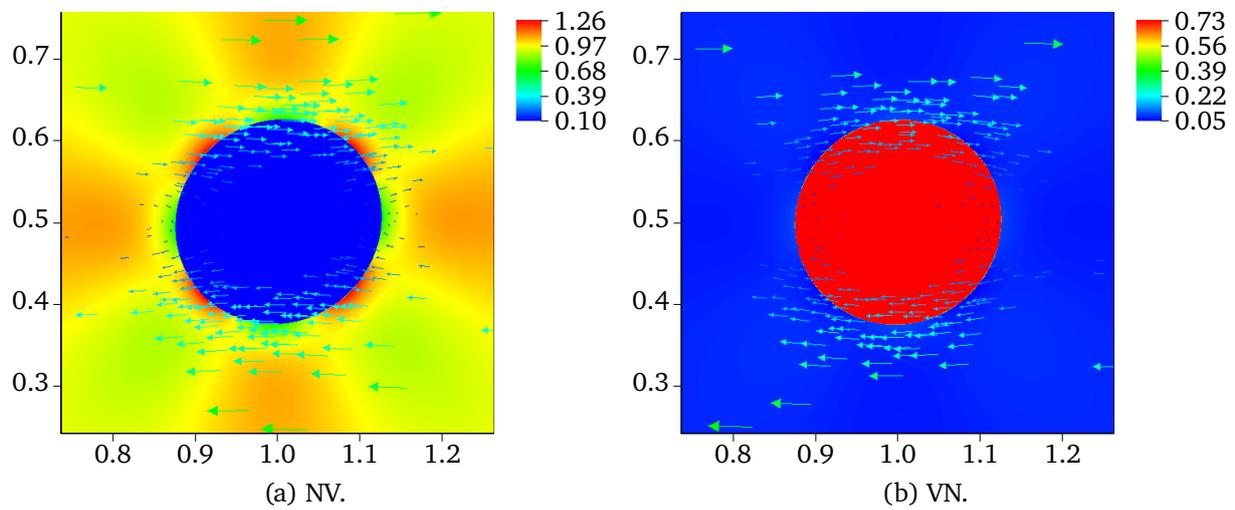


Figure 7.19.: Pseudocolour plot of the shear stress τ_{xy} inside and outside of the droplet for the high-order simulation at $t = 0.020$. The stress jump because of the jump in the Weissenberg number can be seen. The arrows are the velocity vectors.

8. Conclusion

We have presented a new DG discretisation of a governing system of equations for viscoelastic flow. We have also deduced an XDG discretisation for a multiphase viscoelastic system of equations. As a viscoelastic model we use the widespread Oldroyd B model, which is a macroscopic model with a broad range of applications. The discretisation is implemented in a term-by-term fashion such that the chosen model can easily be extended or modified, or other models can be implemented for the extra stress tensor. The implementation of the stress tensor as a local auxiliary variable in an LDG method makes it possible to use it for other applications such as turbulent flow using turbulence models as well.

The discretisation is high-order with a high accuracy and less numerical diffusion than in low-order discretisation methods such as FV. This causes severe problems solving benchmark problems with sharp corners or high, possibly exponential gradients in the stresses within the flow leading to the HWNP. However, since the focus for this implementation is on the simulation of droplets where no such gradients are expected, the software is not optimized for high Weissenberg numbers. If a better adaption is needed in future, a possible solution would be the implementation of the log-conformation formulation introduced by Fattal and Kupferman (2004). The use of the LDG method as a discretisation of a hyperbolic first order system of equations is the reason that there is no need for elliptic stabilization methods, since the fluxes can handle the hyperbolic equations very well. Compared to the DEVSS methods this means that we have fewer dependent variables, thus a smaller operator matrix and less memory and computational effort when solving the system fully coupled. Actually, the use of the elliptic stabilization methods makes it hard to solve the non-linear system coupled and to solve the linear system directly, even by using HPC in combination with high memory nodes. The smaller operator matrix using LDG makes a fully coupled solution possible, which is more robust than a decoupled system. However, for very fine computational meshes combined with high polynomial orders, even when using LDG, the memory requirements of direct solvers may eventually become unfeasible. Therefore, in the long run, iterative solvers are the means of choice, and several iterative methods are currently under development within the BoSSS group for solving large linear systems of equations.

Compared to the classical LDG method developed for Stokes and Navier-Stokes flow, the method used for discretising the viscoelastic system of equations differs in terms of locality. The presence of a convection term within the constitutive equation using a streamline upwinding flux formulation leads to the stress tensor losing its locality. However, without fluxes for the objective terms and without introducing any fluxes at boundaries, most of the local characteristic is kept. Especially in the case of multiphase flow, the flux at the interface is kept as a free Neumann boundary. Thus, the locality of the stress tensor is kept for all terms in the constitutive equations except for the upwinding in the convection term.

For the description of the surface stress on an fluid interface such as a droplet in an ambient

fluid we chose for this work the Boussinesq-Scriven model, which is already more elaborated than a simple isotropic surface stress tensor with a constant surface tension. However, we think that the physics of a viscoelastic surface for the simulation of an oscillating viscoelastic droplet are still not completely displayed and it would be interesting to include characteristics such as elasticity, the memory effect, or the influence of normal stress differences into the surface stress model.

Bibliography

- Aboubacar, M. and Webster, M. F. (2001). “A Cell-Vertex Finite Volume/Element Method on Triangles for Abrupt Contraction Viscoelastic Flows”. In: *Journal of Non-Newtonian Fluid Mechanics* 98.2, pp. 83–106. ISSN: 0377-0257. DOI: 10.1016/S0377-0257(00)00196-8.
- Aboubacar, M. and Webster, M. F. (2003). “Development of an Optimal Hybrid Finite Volume/Element Method for Viscoelastic Flows”. In: *International Journal for Numerical Methods in Fluids* 41.11, pp. 1147–1172. ISSN: 1097-0363. DOI: 10.1002/flid.484.
- Agullo, E., Amestoy, P., Bremond, M., Buttari, A., Combes, P., Fevre, A., and Guermouche, A. (2017). *MUltifrontal Massively Parallel Solver*. Version 5.0.2. Lyon, Inria, Bordeaux: CERFACS, ENS Lyon, INPT(ENSEEIH)-IRIT, Inria and University of Bordeaux.
- Arnold, D., Brezzi, F., Cockburn, B., and Marini, L. (2002). “Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems”. In: *SIAM Journal on Numerical Analysis* 39.5, pp. 1749–1779. ISSN: 0036-1429. DOI: 10.1137/S0036142901384162.
- Arnold, D. N. (1982). “An Interior Penalty Finite Element Method with Discontinuous Elements”. In: *SIAM Journal on Numerical Analysis* 19.4, pp. 742–760. ISSN: 0036-1429. JSTOR: 2157030.
- Baaijens, F. P. T. (1994a). “Application of Low-Order Discontinuous Galerkin Methods to the Analysis of Viscoelastic Flows”. In: *Journal of Non-Newtonian Fluid Mechanics* 52.1, pp. 37–57. ISSN: 0377-0257. DOI: 10.1016/0377-0257(94)85057-7.
- Baaijens, F. P. T. (1994b). “Numerical Experiments with a Discontinuous Galerkin Method Including Monotonicity Enforcement on the Stick-Slip Problem”. In: *Journal of Non-Newtonian Fluid Mechanics* 51.2, pp. 141–159. ISSN: 0377-0257. DOI: 10.1016/0377-0257(94)85009-7.
- Baaijens, F. P. T. (1998). “An Iterative Solver for the DEVSS/DG Method with Application to Smooth and Non-Smooth Flows of the Upper Convected Maxwell Fluid”. In: *Journal of Non-Newtonian Fluid Mechanics* 75.2-3, pp. 119–138. ISSN: 0377-0257. DOI: 10.1016/S0377-0257(97)00086-4.
- Baaijens, F. P. T., Selen, S. H. A., Baaijens, H. P. W., Peters, G. W. M., and Meijer, H. E. H. (1997). “Viscoelastic Flow Past a Confined Cylinder of a Low Density Polyethylene Melt”. In: *Journal of Non-Newtonian Fluid Mechanics* 68.2, pp. 173–203. ISSN: 0377-0257. DOI: [https://doi.org/10.1016/S0377-0257\(96\)01519-4](https://doi.org/10.1016/S0377-0257(96)01519-4).
- Babuška, I. (1973). “The Finite Element Method with Lagrangian Multipliers”. In: *Numerische Mathematik* 20.3, pp. 179–192. ISSN: 0945-3245. DOI: 10.1007/BF01436561.
- Bassi, F. and Rebay, S. (1997). “High-Order Accurate Discontinuous Finite Element Solution of the 2D Euler Equations”. In: *Journal of Computational Physics* 138.2, pp. 251–285. ISSN: 00219991. DOI: 10.1006/jcph.1997.5454.

- Beck, L. (2018). “Numerical Integration over Implicitly Defined Surfaces and Volumes”. Masterarbeit.
- Bodnár, T., Sequeira, A., and Prosi, M. (2011). “On the Shear-Thinning and Viscoelastic Effects of Blood Flow under Various Flow Rates”. In: *Applied Mathematics and Computation. Numerical Analysis of Fluid Flow and Heat Transfer* 217.11, pp. 5055–5067. ISSN: 0096-3003. DOI: 10.1016/j.amc.2010.07.054.
- Bogaerds, A. C. B., Verbeeten, W. M. H., and Baaijens, F. P. T. (2000). “Successes and Failures of Discontinuous Galerkin Methods in Viscoelastic Fluid Analysis”. In: *Discontinuous Galerkin Methods*. Ed. by B. Cockburn, G. E. Karniadakis, and C.-W. Shu. Lecture Notes in Computational Science and Engineering 11. Springer Berlin Heidelberg, pp. 263–270. ISBN: 978-3-642-64098-8 978-3-642-59721-3.
- Bonito, A. and Burman, E. (2008). “A Continuous Interior Penalty Method for Viscoelastic Flows”. In: *SIAM Journal on Scientific Computing* 30.3, pp. 1156–1177. ISSN: 1064-8275. DOI: 10.1137/060677033.
- Bothe, D. and Prüss, J. (2010). “On the Two-Phase Navier–Stokes Equations with Boussinesq–Scriven Surface Fluid”. English. In: *Journal of Mathematical Fluid Mechanics* 12.1. Communicated by G. P. Galdi, pp. 133–150. DOI: 10.1007/s00021-008-0278-x.
- Brezzi, F. (1974). “On the Existence, Uniqueness and Approximation of Saddle-Point Problems Arising from Lagrangian Multipliers”. In: *Revue française d’automatique, informatique, recherche opérationnelle. Analyse numérique* 8.R2, pp. 129–151. ISSN: 0397-9342. DOI: 10.1051/m2an/197408R201291.
- Brooks, A. N. and Hughes, T. J. R. (1982). “Streamline Upwind/Petrov-Galerkin Formulations for Convection Dominated Flows with Particular Emphasis on the Incompressible Navier-Stokes Equations”. In: *Computer Methods in Applied Mechanics and Engineering* 32.1, pp. 199–259. ISSN: 0045-7825. DOI: 10.1016/0045-7825(82)90071-8.
- Castillo, P., Cockburn, B., Perugia, I., and Schötzau, D. (2000). “An A Priori Error Analysis of the Local Discontinuous Galerkin Method for Elliptic Problems”. In: *SIAM Journal on Numerical Analysis* 38.5, pp. 1676–1706. ISSN: 0036-1429, 1095-7170. DOI: 10.1137/S0036142900371003.
- Cesmelioglu, A., Cockburn, B., and Qiu, W. (2017). “Analysis of a Hybridizable Discontinuous Galerkin Method for the Steady-State Incompressible Navier-Stokes Equations”. In: *Mathematics of Computation* 86.306, pp. 1643–1670. ISSN: 0025-5718, 1088-6842. DOI: 10.1090/mcom/3195.
- Chauvière, C. and Owens, R. G. (2001). “A New Spectral Element Method for the Reliable Computation of Viscoelastic Flow”. In: *Computer Methods in Applied Mechanics and Engineering* 190.31, pp. 3999–4018. ISSN: 0045-7825. DOI: 10.1016/S0045-7825(01)00177-3.
- Chauvière, C. and Owens, R. G. (2000). “How Accurate Is Your Solution?: Error Indicators for Viscoelastic Flow Calculations”. In: *Journal of Non-Newtonian Fluid Mechanics* 95.1, pp. 1–33. ISSN: 0377-0257. DOI: 10.1016/S0377-0257(00)00158-0.
- Chinyoka, T., Renardy, Y. Y., Renardy, M., and Khismatullin, D. B. (2005). “Two-Dimensional Study of Drop Deformation under Simple Shear for Oldroyd-B Liquids”. In: *Journal of Non-Newtonian Fluid Mechanics* 130.1, pp. 45–56. ISSN: 0377-0257. DOI: 10.1016/j.jnnfm.2005.07.005.
- Claus, S. and Phillips, T. (2013). “Viscoelastic Flow around a Confined Cylinder Using Spectral/HP Element Methods”. In: *Journal of Non-Newtonian Fluid Mechanics* 200, pp. 131–146. ISSN: 03770257. DOI: 10.1016/j.jnnfm.2013.03.004.

-
- Cockburn, B., Kanschat, G., Schötzau, D., and Schwab, C. (2002). “Local Discontinuous Galerkin Methods for the Stokes System”. In: *SIAM Journal on Numerical Analysis* 40.1, pp. 319–343. ISSN: 0036-1429. DOI: 10.1137/S0036142900380121.
- Cockburn, B., Kanschat, G., and Schötzau, D. (2005a). “A Locally Conservative LDG Method for the Incompressible Navier-Stokes Equations”. In: *Mathematics of Computation* 74.251, pp. 1067–1095. ISSN: 0025-5718, 1088-6842. DOI: 10.1090/S0025-5718-04-01718-1.
- Cockburn, B., Kanschat, G., and Schötzau, D. (2005b). “The Local Discontinuous Galerkin Method for Linearized Incompressible Fluid Flow: A Review”. In: *Computers & Fluids*. Residual Distribution Schemes, Discontinuous Galerkin Schemes and Adaptation 34.4, pp. 491–506. ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2003.08.005.
- Cockburn, B., Kanschat, G., and Schötzau, D. (2007). “A Note on Discontinuous Galerkin Divergence-Free Solutions of the Navier–Stokes Equations”. In: *Journal of Scientific Computing* 31.1, pp. 61–73. ISSN: 1573-7691. DOI: 10.1007/s10915-006-9107-7.
- Cockburn, B., Kanschat, G., and Schötzau, D. (2009). “An Equal-Order DG Method for the Incompressible Navier-Stokes Equations”. In: *Journal of Scientific Computing* 40.1, pp. 188–210. ISSN: 1573-7691. DOI: 10.1007/s10915-008-9261-1.
- Cockburn, B., Karniadakis, G. E., and Shu, C.-W., eds. (2000). *Discontinuous Galerkin Methods*. Red. by M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, and T. Schlick. Vol. 11. Lecture Notes in Computational Science and Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-642-64098-8 978-3-642-59721-3. DOI: 10.1007/978-3-642-59721-3.
- Cockburn, B. and Shu, C.-W. (1989). “TVB Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws. II. General Framework”. In: *Mathematics of Computation* 52.186, pp. 411–435. ISSN: 0025-5718, 1088-6842. DOI: 10.1090/S0025-5718-1989-0983311-4.
- Cockburn, B. and Shu, C.-W. (1991a). “The Runge-Kutta Local Projection P1-Discontinuous-Galerkin Finite Element Method for Scalar Conservation Laws”. In: *Mathematical Modelling and Numerical Analysis* 25.3, pp. 337–361.
- Cockburn, B. and Shu, C.-W. (2001). “Runge–Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems”. In: *Journal of Scientific Computing* 16.3, pp. 173–261. ISSN: 0885-7474, 1573-7691. DOI: 10.1023/A:1012873910884.
- Cockburn, B. and Shu, C.-W. (1998). “The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems”. In: *SIAM Journal on Numerical Analysis* 35.6, pp. 2440–2463. ISSN: 0036-1429. DOI: 10.1137/S0036142997316712.
- Cockburn, B. and Shu, C.-W. (1991b). *The P1-RKDG Method for Two-Dimensional Euler Equations of Gas Dynamics*. NASA Contractor Report 91-32. Hampton: ICASE, NASA Langley Research Center.
- Comminal, R., Spangenberg, J., and Hattel, J. H. (2015). “Robust Simulations of Viscoelastic Flows at High Weissenberg Numbers with the Streamfunction/Log-Conformation Formulation”. In: *Journal of Non-Newtonian Fluid Mechanics* 223, pp. 37–61. ISSN: 0377-0257. DOI: 10.1016/j.jnnfm.2015.05.003.
- Connolly, J. A. D. and Podladchikov, Y. Y. (2000). “Temperature-Dependent Viscoelastic Compaction and Compartmentalization in Sedimentary Basins”. In: *Tectonophysics* 324.3, pp. 137–168. ISSN: 0040-1951. DOI: 10.1016/S0040-1951(00)00084-6.

- Coronado, O. M., Arora, D., Behr, M., and Pasquali, M. (2007). “A Simple Method for Simulating General Viscoelastic Fluid Flows with an Alternate Log-Conformation Formulation”. In: *Journal of Non-Newtonian Fluid Mechanics* 147.3, pp. 189–199. ISSN: 03770257. DOI: 10.1016/j.jnnfm.2007.08.005.
- Crochet, M., Davies, A., and Walters, K. (1984). *Numerical Simulation of Non-Newtonian Flow*. Rheology Series 1. Amsterdam, Oxford, New York, Tokyo: Elsevier. ISBN: 0-444-42291-9.
- De Gennes, P.-G., Brochard-Wyart, F., and Quéré, D. (2004). *Capillarity and Wetting Phenomena*. New York, NY: Springer New York. ISBN: 978-1-4419-1833-8 978-0-387-21656-0. DOI: 10.1007/978-0-387-21656-0.
- Deville, M. O., Fischer, P. F., and Mund, E. H. (2002). *High-Order Methods for Incompressible Fluid Flow*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge: Cambridge University Press. ISBN: 10.1017/CBO9780511546792.
- Di Pietro, D. A. and Ern, A. (2012). *Mathematical Aspects of Discontinuous Galerkin Methods*. Mathématiques et Applications 69. OCLC: ocn777209795. Berlin ; New York: Springer. 384 pp. ISBN: 978-3-642-22979-4.
- Dou, H.-S. and Phan-Thien, N. (2007). “Viscoelastic Flow Past a Confined Cylinder: Instability and Velocity Inflection”. In: *Chemical Engineering Science* 62.15, pp. 3909–3929. ISSN: 00092509. DOI: 10.1016/j.ces.2007.03.040.
- Ebrahimi, M., Tomkovic, T., Liu, G., Doufas, A. A., and Hatzikiriakos, S. G. (2018). “Melt Fracture of Linear Low-Density Polyethylenes: Die Geometry and Molecular Weight Characteristics”. In: *Physics of Fluids* 30.5, p. 053103. ISSN: 1070-6631, 1089-7666. DOI: 10.1063/1.5029380.
- Fan, Y., Yang, H., and Tanner, R. I. (2005). “Stress Boundary Layers in the Viscoelastic Flow Past a Cylinder in a Channel: Limiting Solutions”. In: *Acta Mechanica Sinica* 21.4, pp. 311–321. ISSN: 1614-3116. DOI: 10.1007/s10409-005-0040-z.
- Fattal, R. and Kupferman, R. (2004). “Constitutive Laws for the Matrix-Logarithm of the Conformation Tensor”. In: *Journal of Non-Newtonian Fluid Mechanics* 123.2–3, pp. 281–285. ISSN: 0377-0257. DOI: 10.1016/j.jnnfm.2004.08.008.
- Fattal, R. and Kupferman, R. (2005). “Time-Dependent Simulation of Viscoelastic Flows at High Weissenberg Number Using the Log-Conformation Representation”. In: *Journal of Non-Newtonian Fluid Mechanics* 126.1, pp. 23–37. ISSN: 03770257. DOI: 10.1016/j.jnnfm.2004.12.003.
- Ferziger, J. H. and Perić, M. (2008). *Numerische Strömungsmechanik*. Trans. by K. Perić. OCLC: 244010653. Berlin Heidelberg: Springer. 509 pp. ISBN: 978-3-540-67586-0 978-3-540-68228-8.
- Fiétier, N. and Deville, M. O. (2003). “Time-Dependent Algorithms for the Simulation of Viscoelastic Flows with Spectral Element Methods: Applications and Stability”. In: *Journal of Computational Physics* 186.1, pp. 93–121. ISSN: 0021-9991. DOI: 10.1016/S0021-9991(03)00013-5.
- Fortin, A., Béliveau, A., Heuzey, M. C., and Lioret, A. (2000). “Ten Years Using Discontinuous Galerkin Methods for Polymer Processing Problems”. In: pp. 321–326. DOI: 10.1007/978-3-642-59721-3_28.
- Fortin, A. and Fortin, M. (1990). “A Preconditioned Generalized Minimal Residual Algorithm for the Numerical Solution of Viscoelastic Fluid Flows”. In: *Journal of Non-Newtonian Fluid Mechanics* 36, pp. 277–288. ISSN: 0377-0257. DOI: 10.1016/0377-0257(90)85014-P.

-
- Fortin, A. and Zine, A. (1992). “An Improved GMRES Method for Solving Viscoelastic Fluid Flow Problems”. In: *Journal of Non-Newtonian Fluid Mechanics* 42.1, pp. 1–18. ISSN: 0377-0257. DOI: 10.1016/0377-0257(92)80001-E.
- Fortin, A., Zine, A., and Agassant, J.-F. (1992). “Computing Viscoelastic Fluid Flow Problems at Low Cost”. In: *Journal of Non-Newtonian Fluid Mechanics* 45.2, pp. 209–229. ISSN: 0377-0257. DOI: 10.1016/0377-0257(92)85004-G.
- Fortin, M. and Fortin, A. (1989). “A New Approach for the FEM Simulation of Viscoelastic Flows”. In: *Journal of Non-Newtonian Fluid Mechanics* 32.3, pp. 295–310. ISSN: 0377-0257. DOI: 10.1016/0377-0257(89)85012-8.
- Gava, A. and Lucchetta, G. (2012). “On the Performance of a Viscoelastic Constitutive Model for Micro Injection Moulding Simulations”. In: *Express Polymer Letters* 6.5, pp. 417–426. ISSN: 1788618X. DOI: 10.3144/expresspolymlett.2012.44.
- Giesekus, H. (1994). *Phänomenologische Rheologie: Eine Einführung*. Springer Berlin Heidelberg.
- Girault, V., Rivière, B., and Wheeler, M. (2005). “A Discontinuous Galerkin Method with Nonoverlapping Domain Decomposition for the Stokes and Navier-Stokes Problems”. In: *Mathematics of Computation* 74.249, pp. 53–84. ISSN: 0025-5718, 1088-6842. DOI: 10.1090/S0025-5718-04-01652-7.
- Gross, S. and Reusken, A. (2011). *Numerical Methods for Two-Phase Incompressible Flows*. Springer Science & Business Media. 487 pp. ISBN: 978-3-642-19686-7. Google Books: YVIM2CviC5QC.
- Guénette, R. and Fortin, M. (1995). “A New Mixed Finite Element Method for Computing Viscoelastic Flows”. In: *Journal of Non-Newtonian Fluid Mechanics* 60.1, pp. 27–52. ISSN: 0377-0257. DOI: 10.1016/0377-0257(95)01372-3.
- Heimann, F., Engwer, C., Ippisch, O., and Bastian, P. (2013). “An Unfitted Interior Penalty Discontinuous Galerkin Method for Incompressible Navier–Stokes Two-Phase Flow”. In: *International Journal for Numerical Methods in Fluids* 71.3, pp. 269–293. ISSN: 1097-0363. DOI: 10.1002/flid.3653.
- Hesthaven, J. S. and Warburton, T. (2008). *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. Texts in Applied Mathematics. New York: Springer-Verlag. ISBN: 978-0-387-72065-4. DOI: 10.1007/978-0-387-72067-8.
- Hu, H.-W. and Granick, S. (1992). “Viscoelastic Dynamics of Confined Polymer Melts”. In: *Science* 258.5086, pp. 1339–1342. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.258.5086.1339. pmid: 17778360.
- Hulsen, M. A., Fattal, R., and Kupferman, R. (2005). “Flow of Viscoelastic Fluids Past a Cylinder at High Weissenberg Number: Stabilized Simulations Using Matrix Logarithms”. In: *Journal of Non-Newtonian Fluid Mechanics* 127.1, pp. 27–39. ISSN: 03770257. DOI: 10.1016/j.jnnfm.2005.01.002.
- Joseph, D. D. and Saut, J. C. (1986). “Change of Type and Loss of Evolution in the Flow of Viscoelastic Fluids”. In: *Journal of Non-Newtonian Fluid Mechanics* 20, pp. 117–141. ISSN: 0377-0257. DOI: 10.1016/0377-0257(86)80018-0.
- Joseph, D. D. (1990). *Fluid Dynamics of Viscoelastic Liquids*. Vol. 84. Applied Mathematical Sciences. New York [u.a.]: Springer. ISBN: 0-387-97155-6.
- Joseph, D. D., Renardy, M., and Saut, J.-C. (1987). “Hyperbolicity and Change of Type in the Flow of Viscoelastic Fluids”. In: *Analysis and Thermomechanics*. Springer Berlin Heidelberg, pp. 25–63. ISBN: 978-3-540-18125-5 978-3-642-61598-6.

- Keith, B., Knechtges, P., Roberts, N., Elgeti, S., Behr, M., and Demkowicz, L. (2017). “An Ultraweak DPG Method for Viscoelastic Fluids”. In: *Journal of Non-Newtonian Fluid Mechanics* 247, pp. 107–122. ISSN: 03770257. DOI: 10.1016/j.jnnfm.2017.06.006.
- Kelley, C. (1995). *Iterative Methods for Linear and Nonlinear Equations*. Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics. 169 pp. ISBN: 978-0-89871-352-7. DOI: 10.1137/1.9781611970944.
- Ketata, M., Ayadi, A., Elkissi, N., and Bradai, C. (2017). “Effect of Rheological and Physical Properties on Mitigation of Melt Fracture Instability during Extrusion of Polymer Melts through a Radial Flow Die”. In: *Rheologica Acta* 56.4, pp. 341–350. ISSN: 0035-4511, 1435-1528. DOI: 10.1007/s00397-017-0995-2.
- Kikker, A. and Kummer, F. (2018). “A High-Order Local Discontinuous Galerkin Scheme for Viscoelastic Fluid Flow”. In: *Recent Advances in Computational Engineering*. Ed. by M. Schäfer, M. Behr, M. Mehl, and B. Wohlmuth. Lecture Notes in Computational Science and Engineering. Springer International Publishing, pp. 51–61. ISBN: 978-3-319-93891-2.
- Kikker, A., Kummer, F., and Oberlack, M. (2020). “A Fully Coupled High-Order Discontinuous Galerkin Solver for Viscoelastic Fluid Flow”. In: *International Journal for Numerical Methods in Fluids (under review)*.
- Kim, J. M., Kim, C., Ahn, K. H., and Lee, S. J. (2004). “An Efficient Iterative Solver and High-Resolution Computations of the Oldroyd-B Fluid Flow Past a Confined Cylinder”. In: *Journal of Non-Newtonian Fluid Mechanics* 123.2, pp. 161–173. ISSN: 0377-0257. DOI: 10.1016/j.jnnfm.2004.08.003.
- Kim, J. M., Kim, C., Kim, J. H., Chung, C., Ahn, K. H., and Lee, S. J. (2005). “High-Resolution Finite Element Simulation of 4:1 Planar Contraction Flow of Viscoelastic Fluid”. In: *Journal of Non-Newtonian Fluid Mechanics* 129.1, pp. 23–37. ISSN: 0377-0257. DOI: 10.1016/j.jnnfm.2005.04.007.
- King, R. C., Apelian, M. R., Armstrong, R. C., and Brown, R. A. (1988). “Numerically Stable Finite Element Techniques for Viscoelastic Calculations in Smooth and Singular Geometries”. In: *Journal of Non-Newtonian Fluid Mechanics* 29, pp. 147–216. ISSN: 0377-0257. DOI: 10.1016/0377-0257(88)85054-7.
- Klajj, C. M., van der Vegt, J. J. W., and van der Ven, H. (2006). “Space–Time Discontinuous Galerkin Method for the Compressible Navier–Stokes Equations”. In: *Journal of Computational Physics* 217.2, pp. 589–611. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2006.01.018.
- Kummer, F., Müller, B., Geisenhofer, M., Kahle, J., Krause, D., Smuda, M., Utz, T., Krämer-Eis, S., Dierkes, D., Kikker, A., and Keil, M. (2020a). *BoSSS-Handbook.Pdf*. Version 509.
- Kummer, F. and Oberlack, M. (2013). “An Extension of the Discontinuous Galerkin Method for the Singular Poisson Equation”. In: *SIAM Journal on Scientific Computing* 35.2, A603–A622. ISSN: 1064-8275. DOI: 10.1137/120878586.
- Kummer, F. (2012). “The BoSSS Discontinuous Galerkin Solver for Incompressible Fluid Dynamics and an Extension to Singular Equations.” Darmstadt: TU Darmstadt. 160 pp.
- Kummer, F. (2017). “Extended Discontinuous Galerkin Methods for Two-Phase Flows: The Spatial Discretization”. In: *International Journal for Numerical Methods in Engineering* 109.2, pp. 259–289. ISSN: 00295981. DOI: 10.1002/nme.5288.
- Kummer, F., Müller, B., and Utz, T. (2018). “Time Integration for Extended Discontinuous Galerkin Methods with Moving Domains”. In: *International Journal for Numerical Methods in Engineering* 113.5, pp. 767–788. ISSN: 1097-0207. DOI: 10.1002/nme.5634.

-
- Kummer, F., Smuda, M., and Weber, J. (2020b). “BoSSS: A Package for Multigrid Extended Discontinuous Galerkin Methods”. In: arXiv: 2003.02431 [cs, math].
- Kummer, F. and Warburton, T. (2016). “Patch-Recovery Filters for Curvature in Discontinuous Galerkin-Based Level-Set Methods”. In: *Communications in Computational Physics* 19.2, pp. 329–353. ISSN: 1815-2406, 1991-7120. DOI: 10.4208/cicp.191114.140715a.
- Kupferman, R. (2005). “On the Linear Stability of Plane Couette Flow for an Oldroyd-B Fluid and Its Numerical Approximation”. In: *Journal of Non-Newtonian Fluid Mechanics* 127.2-3, pp. 169–190. ISSN: 03770257. DOI: 10.1016/j.jnnfm.2005.03.002.
- Kynch, R. and Phillips, T. (2017). “A High Resolution Spectral Element Approximation of Viscoelastic Flows in Axisymmetric Geometries Using a DEVSS-G/DG Formulation”. In: *Journal of Non-Newtonian Fluid Mechanics* 240, pp. 15–33. ISSN: 03770257. DOI: 10.1016/j.jnnfm.2016.12.008.
- Lehrenfeld, C. and Reusken, A. (2012). “Nitsche-XFEM with Streamline Diffusion Stabilization for a Two-Phase Mass Transport Problem”. In: *SIAM Journal on Scientific Computing* 34.5, A2740–A2759. ISSN: 1064-8275. DOI: 10.1137/110855235.
- Lehrenfeld, C. and Reusken, A. (2013). “Analysis of a Nitsche XFEM-DG Discretization for a Class of Two-Phase Mass Transport Problems”. In: *SIAM Journal on Numerical Analysis* 51.2, pp. 958–983. ISSN: 0036-1429. DOI: 10.1137/120875260.
- Lehrenfeld, C. (2015a). “On a Space-Time Extended Finite Element Method for the Solution of a Class of Two-Phase Mass Transport Problems”. RWTH Aachen.
- Lehrenfeld, C. (2015b). “The Nitsche XFEM-DG Space-Time Method and Its Implementation in Three Space Dimensions”. In: *SIAM Journal on Scientific Computing* 37.1, A245–A270. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/130943534. arXiv: 1408.2941.
- Lehrenfeld, C. (2016). “High Order Unfitted Finite Element Methods on Level Set Domains Using Isoparametric Mappings”. In: *Computer Methods in Applied Mechanics and Engineering* 300, pp. 716–733. ISSN: 0045-7825. DOI: 10.1016/j.cma.2015.12.005.
- Lesaint, P. and Raviart, P. (1974). “On a Finite Element Method for Solving the Neutron Transport Equation”. In: *Mathematical Aspects of Finite Elements in Partial Differential Equations*. Elsevier, pp. 89–123. ISBN: 978-0-12-208350-1. DOI: 10.1016/B978-0-12-208350-1.50008-X.
- Li, B. Q. (2006). *Discontinuous Finite Elements in Fluid Dynamics and Heat Transfer*. Computational Fluid and Solid Mechanics. London: Springer. 578 pp. ISBN: 978-1-85233-988-3 978-1-84628-205-8.
- Li, Q., Ouyang, J., Yang, B., and Jiang, T. (2011). “Modelling and Simulation of Moving Interfaces in Gas-Assisted Injection Moulding Process”. In: *Applied Mathematical Modelling* 35.1, pp. 257–275. ISSN: 0307-904X. DOI: 10.1016/j.apm.2010.06.002.
- Marchal, J. M. and Crochet, M. J. (1987). “A New Mixed Finite Element for Calculating Viscoelastic Flow”. In: *Journal of Non-Newtonian Fluid Mechanics* 26.1, pp. 77–114. ISSN: 0377-0257. DOI: 10.1016/0377-0257(87)85048-6.
- McKinley, G. H., Armstrong, Robert C., and Brown, R. (1993). “The Wake Instability in Viscoelastic Flow Past Confined Circular Cylinders”. In: *z* 344.1671, pp. 265–304. DOI: 10.1098/rsta.1993.0091.
- Mirzakhaili, E. and Nejat, A. (2015). “High-Order Solution of Viscoelastic Fluids Using the Discontinuous Galerkin Method”. In: *Journal of Fluids Engineering-Transactions of the Asme* 137.3. WOS:000352431700011, p. 031205. ISSN: 0098-2202. DOI: 10.1115/1.4028779.

-
- Mu, Y., Chen, A., Zhao, G., Cui, Y., Feng, J., and Ren, F. (2019). “Finite Element Simulation of Three-Dimensional Viscoelastic Flow at High Weissenberg Number Based on the Log-Conformation Formulation”. In: *Mechanics of Time-Dependent Materials* 23.4, pp. 477–495. ISSN: 1573-2738. DOI: 10.1007/s11043-018-9401-4.
- Niethammer, M., Marschall, H., Kunkelmann, C., and Bothe, D. (2018). “A numerical stabilization framework for viscoelastic fluid flow using the finite volume method on general unstructured meshes”. In: *International Journal for Numerical Methods in Fluids* 86.2, pp. 131–166. DOI: 10.1002/flid.4411. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/flid.4411>.
- Noll, W. (1958). “A Mathematical Theory of the Mechanical Behavior of Continuous Media”. In: *Archive for Rational Mechanics and Analysis* 2.1, pp. 197–226. ISSN: 0003-9527, 1432-0673. DOI: 10.1007/BF00277929.
- Oldroyd, J. G. and Wilson, A. H. (1950). “On the Formulation of Rheological Equations of State”. In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 200.1063, pp. 523–541. DOI: 10.1098/rspa.1950.0035.
- Oliveira, P. J. and Miranda, A. I. P. (2005). “A Numerical Study of Steady and Unsteady Viscoelastic Flow Past Bounded Cylinders”. In: *Journal of Non-Newtonian Fluid Mechanics* 127.1, pp. 51–66. ISSN: 0377-0257. DOI: 10.1016/j.jnnfm.2005.02.003.
- Owens, R. G. and Phillips, T. N. (2002). *Computational Rheology*. London; River Edge, NJ: Imperial College Press ; Distributed by World Scientific Pub. Co. ISBN: 978-1-86094-942-5 1-86094-942-8.
- Owens, R. G., Chauvière, C., and Philips, T. N. (2002). “A Locally-Upwinded Spectral Technique (LUST) for Viscoelastic Flows”. In: *Journal of Non-Newtonian Fluid Mechanics. Numerical Methods Workshop S.I.* 108.1, pp. 49–71. ISSN: 0377-0257. DOI: 10.1016/S0377-0257(02)00124-6.
- Persson, P.-O. and Peraire, J. (2006). “Sub-Cell Shock Capturing for Discontinuous Galerkin Methods”. In: *44th AIAA Aerospace Sciences Meeting and Exhibit. 44th AIAA Aerospace Sciences Meeting and Exhibit*. Reno, Nevada: American Institute of Aeronautics and Astronautics. ISBN: 978-1-62410-039-0. DOI: 10.2514/6.2006-112.
- Phan-Thien, N. and Mai-Duy, N. (2017). *Understanding Viscoelasticity*. 3rd ed. New York, NY: Springer Berlin Heidelberg. ISBN: 978-3-319-61999-6.
- Phillips, T. N. and Williams, A. J. (1999). “Viscoelastic Flow through a Planar Contraction Using a Semi-Lagrangian Finite Volume Method”. In: *Journal of Non-Newtonian Fluid Mechanics* 87.2–3, pp. 215–246. ISSN: 0377-0257. DOI: 10.1016/S0377-0257(99)00065-8.
- Pilitsis, S., Souvaliotis, A., and Beris, A. N. (1991). “Viscoelastic Flow in a Periodically Constricted Tube: The Combined Effect of Inertia, Shear Thinning, and Elasticity”. In: *Journal of Rheology* 35.4, pp. 605–646. ISSN: 0148-6055. DOI: 10.1122/1.550183.
- Rajagopalan, D., Armstrong, R. C., and Brown, R. A. (1990). “Finite Element Method for Calculation of Steady, Viscoelastic Flow Using Constitutive Equations with a Newtonian Viscosity”. In: *Journal of Non-Newtonian Fluid Mechanics* 36, pp. 159–192. ISSN: 0377-0257. DOI: 10.1016/0377-0257(90)85008-M.
- Reed, W. and Hill, T. (1973). *Triangular Mesh Methods for the Neutron Transport Equation*. Los Alamos Scientific Lab., Los Alamos, New Mexico (USA).
- Renardy, M. (1985). “Existence of Slow Steady Flows of Viscoelastic Fluids with Differential Constitutive Equations”. In: *ZAMM - Journal of Applied Mathematics and Mechanics /*

- Zeitschrift für Angewandte Mathematik und Mechanik* 65.9, pp. 449–451. ISSN: 1521-4001. DOI: 10.1002/zamm.19850650919.
- Rhebergen, S., Cockburn, B., and van der Vegt, J. J. W. (2013). “A Space–Time Discontinuous Galerkin Method for the Incompressible Navier–Stokes Equations”. In: *Journal of Computational Physics* 233, pp. 339–358. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2012.08.052.
- Sauerland, H. and Fries, T.-P. (2011). “The Extended Finite Element Method for Two-Phase and Free-Surface Flows: A Systematic Study”. In: *Journal of Computational Physics* 230.9, pp. 3369–3390. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2011.01.033.
- Sauerland, H. and Fries, T.-P. (2013). “The Stable XFEM for Two-Phase Flows”. In: *Computers & Fluids*. USNCCM Moving Boundaries 87, pp. 41–49. ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2012.10.017.
- Saye, R. I. (2015). “High-Order Quadrature Methods for Implicitly Defined Surfaces and Volumes in Hyperrectangles”. In: *SIAM Journal on Scientific Computing* 37.2, A993–A1019. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/140966290.
- Schenk, O. and Gärtner, K. (2002). “Two-level dynamic scheduling in PARDISO: Improved scalability on shared memory multiprocessing systems”. In: *Parallel Computing* 28.2, pp. 187–197. ISSN: 0167-8191. DOI: [https://doi.org/10.1016/S0167-8191\(01\)00135-1](https://doi.org/10.1016/S0167-8191(01)00135-1).
- Schiesser, W. E. (2012). *The Numerical Method of Lines: Integration of Partial Differential Equations*. Elsevier. 341 pp. ISBN: 978-0-12-801551-3. Google Books: 2YDNCgAAQBAJ.
- Shahbazi, K. (2005). “An Explicit Expression for the Penalty Parameter of the Interior Penalty Method”. In: *Journal of Computational Physics* 205.2, pp. 401–407. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2004.11.017.
- Shahbazi, K., Fischer, P. F., and Ethier, C. R. (2007). “A High-Order Discontinuous Galerkin Method for the Unsteady Incompressible Navier–Stokes Equations”. In: *Journal of Computational Physics* 222.1, pp. 391–407. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2006.07.029.
- Sonntag, M. and Munz, C.-D. (2017). “Efficient Parallelization of a Shock Capturing for Discontinuous Galerkin Methods Using Finite Volume Sub-Cells”. In: *Journal of Scientific Computing* 70.3, pp. 1262–1289. ISSN: 1573-7691. DOI: 10.1007/s10915-016-0287-5.
- Sun, J., Smith, M. D., Armstrong, R. C., and Brown, R. A. (1999). “Finite Element Method for Viscoelastic Flows Based on the Discrete Adaptive Viscoelastic Stress Splitting and the Discontinuous Galerkin Method: DAVSS-G/DG”. In: *Journal of Non-Newtonian Fluid Mechanics* 86.3, pp. 281–307. ISSN: 0377-0257. DOI: 10.1016/S0377-0257(98)00176-1.
- Sun, J., Phan-Thien, N., and Tanner, R. I. (1996). “An Adaptive Viscoelastic Stress Splitting Scheme and Its Applications: AVSS/SI and AVSS/SUPG”. In: *Journal of Non-Newtonian Fluid Mechanics* 65.1, pp. 75–91. ISSN: 0377-0257. DOI: 10.1016/0377-0257(96)01448-6.
- Sureshkumar, R. and Beris, A. N. (1995). “Effect of Artificial Stress Diffusivity on the Stability of Numerical Calculations and the Flow Dynamics of Time-Dependent Viscoelastic Flows”. In: *Journal of Non-Newtonian Fluid Mechanics* 60.1, pp. 53–80. ISSN: 0377-0257. DOI: 10.1016/0377-0257(95)01377-8.
- Sussman, M. and Hussaini, M. Y. (2003). “A Discontinuous Spectral Element Method for the Level Set Equation”. In: *Journal of Scientific Computing* 19.1, pp. 479–500. ISSN: 1573-7691. DOI: 10.1023/A:1025328714359.

-
- Talwar, K. K. and Khomami, B. (1992). “Accuracy and Convergence of the P- and Hp-Type Finite Element Methods for the Navier-Stokes Equation”. In: *AIChE Journal* 38.1, pp. 83–92. ISSN: 1547-5905. DOI: 10.1002/aic.690380109.
- Thurston, G. (1972). “Viscoelasticity of Human Blood”. In: *Biophysical Journal* 12.9, pp. 1205–1217. ISSN: 00063495. DOI: 10.1016/S0006-3495(72)86156-3.
- Utz, T. and Kummer, F. (2018). “A High-Order Discontinuous Galerkin Method for Extension Problems”. In: *International Journal for Numerical Methods in Fluids* 86.8, pp. 509–518. ISSN: 1097-0363. DOI: 10.1002/flid.4464.
- Utz, T., Kummer, F., and Oberlack, M. (2017). “Interface-Preserving Level-Set Reinitialization for DG-FEM”. In: *International Journal for Numerical Methods in Fluids* 84.4, pp. 183–198. ISSN: 1097-0363. DOI: 10.1002/flid.4344.
- Varchanis, S., Syrakos, A., Dimakopoulos, Y., and Tsamopoulos, J. (2019). “A New Finite Element Formulation for Viscoelastic Flows: Circumventing Simultaneously the LBB Condition and the High-Weissenberg Number Problem”. In: *Journal of Non-Newtonian Fluid Mechanics* 267, pp. 78–97. ISSN: 0377-0257. DOI: 10.1016/j.jnnfm.2019.04.003.
- Wapperom, P. and Webster, M. F. (1998). “A Second-Order Hybrid Finite-Element/Volume Method for Viscoelastic flows1Dedicated to Professor Marcel J. Crochet on the Occasion of His 60th Birthday.1”. In: *Journal of Non-Newtonian Fluid Mechanics* 79.2, pp. 405–431. ISSN: 0377-0257. DOI: 10.1016/S0377-0257(98)00124-4.
- Wu, T.-M. (2005). “A Study of Convergence on the Newton-Homotopy Continuation Method”. In: *Applied Mathematics and Computation* 168.2, pp. 1169–1174. ISSN: 0096-3003. DOI: 10.1016/j.amc.2003.10.068.
- Xue, S. -C., Phan-Thien, N., and Tanner, R. I. (1995). “Numerical Study of Secondary Flows of Viscoelastic Fluid in Straight Pipes by an Implicit Finite Volume Method”. In: *Journal of Non-Newtonian Fluid Mechanics* 59.2, pp. 191–213. ISSN: 0377-0257. DOI: 10.1016/0377-0257(95)01365-3.

A. BoSSSpad-Worksheets

A.1. Validation Local Discontinuous Galerkin Implementation

```
1:      restart

2: // using the Rheology Solver
3: using BoSSS.Application.Rheology;
4: using BoSSS.Solution.AdvancedSolvers;

5: // Workflow Manager Name for Postprocessing
6: WorkflowMgm.Init("LDG_NS_Convergence_Study");

7: // Database
8: var myDb = OpenOrCreateDatabase(@"\\Path\to\Database\LDG_NS_Convergence_Study"
    );

9: // Batch for caculation on the HPC at FDY
10: //var myBatch = new MsHPC2012Client(@"\\hpccluster\hpccluster-scratch\NAME\
    deploy_dir\", "hpccluster", ComputeNodes : new[] {"hpccluster"});
11:
12: //Batch for caculation on local PC
13: var myBatch = new MiniBatchProcessorClient(@"\\Path\to\deploy\directory\
    LDG_NS_Convergence_Study");

14: // Start MiniBatchProcessor
15: MiniBatchProcessor.Server.StartIfNotRunning();

16: // =====
17: // Convergence Parameters
18: // =====

19: int[] pOrder = new int[] {1, 2, 3, 4, 5};
20: int[] Gridlevel = new int[] { 2, 3, 4, 5};

21: // =====
22: // Create grids
23: // =====
```

```

24: int numE = Gridlevel.Length;
25: double L = 10;
26: double H = 1;
27: GridCommons[] grids = new GridCommons[Gridlevel.Length];
28: for (int k = 0; k < numE; k++) {
29:
30:     double h      = Math.Pow(2, -(double)Gridlevel[k] + 1);
31:     double cells  = 1 / h;
32:     int cells2   = (int)cells;
33:
34:     double[] xNodes = GenericBlas.Linspace(-1, 1, cells2 + 1);
35:     double[] yNodes = GenericBlas.Linspace(-1, 1, cells2 + 1);
36:     GridCommons grd;
37:     grd = Grid2D.Cartesian2DGrid(xNodes, yNodes);
38:
39:     grd.EdgeTagNames.Add(1, "Velocity_inlet");
40:
41:     grd.DefineEdgeTags(delegate (double[] X) {
42:         byte et = 0;
43:         if(Math.Abs(X[1] + 1) <= 1.0e-8)
44:             et = 1;
45:         if(Math.Abs(X[1] - 1) <= 1.0e-8)
46:             et = 1;
47:         if (Math.Abs(X[0] + 1) <= 1.0e-8)
48:             et = 1;
49:         if (Math.Abs(X[0] - 1) <= 1.0e-8)
50:             et = 1;
51:
52:         return et;
53:     });
54:
55:     myDb.SaveGrid(ref grd);
56:     grids[k] = grd;
57: }

```

```

58: // =====
59: // physical parameters
60: // =====

```

```

61: double beta          = 0;
62: double Reynolds      = 1;
63: double Weissenberg   = 0.0; //aim Weissenberg number
64: bool RaiseWeissenberg = false;
65: double WeissenbergIncrement = 0.1;

```

```

66: // =====
67: // manufactured solution
68: // =====

```

```

69: static class InitialValues {
70:
71:     static double beta          = 0;
72:     static double Reynolds      = 1;
73:     static double Weissenberg   = 0;
74:

```

```

75:     public static double VelocityXfunction(double[] X) {
76:         return -Math.Exp(X[0]) * (X[1] * Math.Cos(X[1]) + Math.Sin(X[1]))
;
77:     }
78:
79:     public static double VelocityYfunction(double[] X) {
80:         return Math.Exp(X[0]) * X[1] * Math.Sin(X[1]);
81:     }
82:
83:     public static double Pressurefunction(double[] X) {
84:         return 2 * Math.Exp(X[0]) * Math.Sin(X[1]);
85:     }
86:
87:     public static double StressXXfunction(double[] X) {
88:         return -2 * (1 - beta) * Math.Exp(X[0]) * (X[1] * Math.Cos(X[1])
+ Math.Sin(X[1]));
89:     }
90:
91:     public static double StressXYfunction(double[] X) {
92:         return -2 * (1 - beta) * Math.Exp(X[0]) * (Math.Cos(X[1]) - X[1]
* Math.Sin(X[1]));
93:     }
94:
95:     public static double StressYYfunction(double[] X) {
96:         return 2 * (1 - beta) * Math.Exp(X[0]) * (X[1] * Math.Cos(X[1]) +
Math.Sin(X[1]));
97:     }
98:     public static double Phi(double[] X) {
99:         return -1.0;
100:    }
101:    public static double GravityXfunction(double[] X) {
102:        return -1 / Reynolds * Math.Exp(X[0]) * (Math.Exp(X[0]) * Math.
Cos(X[1]) * Math.Cos(X[1]) * Reynolds
103:        - Math.Exp(X[0]) * Reynolds * X[1] * X[1] - Math.Exp(X[0]) *
Reynolds - 2 * Math.Sin(X[1]) * Reynolds + 2 * Math.Sin(X[1]));
104:    }
105:    public static double GravityYfunction(double[] X) {
106:        return 2 * Math.Exp(X[0]) * Math.Cos(X[1]) * ((Reynolds - 1) /
Reynolds);
107:    }
108:    public static double GravityXXfunction(double[] X) {
109:        return 2 * Math.Exp(2 * X[0]) * Weissenberg * (-2 * Math.Cos(X
[1]) * Math.Sin(X[1]) * beta * X[1] * Weissenberg
110:        + 3 * Math.Cos(X[1]) * Math.Cos(X[1]) * beta + 2 * Math.Cos(X[1])
* Math.Sin(X[1]) * X[1] + beta * X[1] * X[1]
111:        - 3 * Math.Cos(X[1]) * Math.Cos(X[1]) - X[1] * X[1] + beta - 1);
112:    }
113:    public static double GravityXYfunction(double[] X) {
114:        return -2 * Math.Exp(2 * X[0]) * (beta - 1) * Weissenberg * (2 *
Math.Pow(Math.Cos(X[1]), 2) * X[1] + 3 * Math.Cos(X[1]) * Math.Sin(X[1]) + X
[1]);
115:    }
116:    public static double GravityYYfunction(double[] X) {
117:        return -2 * Math.Exp(2 * X[0]) * Weissenberg * (-2 * Math.Cos(X
[1]) * Math.Sin(X[1]) * beta * X[1]
118:        + 3 * Math.Cos(X[1]) * Math.Cos(X[1]) * beta + 2 * Math.Cos(X[1])
* Math.Sin(X[1]) * X[1] - 3 * beta * X[1] * X[1]

```

```
119:         - 3 * Math.Cos(X[1]) * Math.Cos(X[1]) + 3 * X[1] * X[1] - 3 *
    beta + 3);
120:     }
121: }
```

```
122: // =====
123: // setup control object for a solver run
124: // =====
```

```
125: List<RheologyControl> Controls = new List<RheologyControl>();
```

```
126: Controls.Clear();
127: foreach(int degree in pOrder) {
128:     int elemInd = 2;
129:     foreach(GridCommons grd in grids) {
130:         RheologyControl C = new RheologyControl();
131:
132:         // Solver Options
133:         C.savetodb = true;
134:         C.DbPath = myDb.Path;
135:         C.SessionName = "Degree" + degree + ", GridLevel" + Gridlevel;
136:         C.ProjectName = "ConvStudyLDG";
137:         C.SetGrid(grd);
138:         C.SetDGdegree(degree);
139:
140:         //Advanced Solvers
141:         C.NonLinearSolver.SolverCode = NonLinearSolverCode.Newton;
142:         C.NonLinearSolver.MaxSolverIterations = 10;
143:         C.NonLinearSolver.MinSolverIterations = 1;
144:         C.NonLinearSolver.ConvergenceCriterion = 1E-7;
145:         C.useFDJacobianForOperatorMatrix = false;
146:
147:         C.LinearSolver.SolverCode = LinearSolverCode.
    classic_pardiso;
148:         C.LinearSolver.MaxSolverIterations = 3;
149:         C.LinearSolver.MinSolverIterations = 1;
150:         C.LinearSolver.ConvergenceCriterion = 5E-7;
151:
152:         //Timestepping
153:         C.NoOfTimesteps = 1;
154:         C.dt = 1E20;
155:         C.dtMax = C.dt;
156:         C.dtMin = C.dt;
157:         C.Timestepper_Scheme = RheologyControl.TimesteppingScheme.
    ImplicitEuler;
158:
159:         //Configuration Shock capturing and body forces
160:         C.UsePerssonSensor = false;
161:         C.SensorLimit = 1e-4;
162:         C.AdaptiveMeshRefinement = false;
163:         C.RefinementLevel = 10;
164:         C.UseArtificialDiffusion = false;
165:         C.Bodyforces = false;
166:
167:         //Debugging and Solver Analysis
168:         C.OperatorMatrixAnalysis = false;
```

```

169:         C.SkipSolveAndEvaluateResidual = false;
170:         C.SetInitialConditions           = true;
171:         C.SetInitialPressure             = true;
172:         C.SetParamsAnalyticalSol        = false;
173:         C.ComputeL2Error                 = true;
174:
175:         //Physical Params
176:         C.Stokes                         = false;
177:         C.FixedStreamwisePeriodicBC     = false;
178:         C.GravitySource                  = true;
179:         C.beta                           = beta;
180:         C.Reynolds                       = Reynolds;
181:         C.Weissenberg                    = Weissenberg; //aim Weissenberg
number!
182:         C.RaiseWeissenberg              = RaiseWeissenberg;
183:         C.WeissenbergIncrement          = WeissenbergIncrement;
184:
185:         //Penalties
186:         C.ViscousPenaltyScaling = 1;
187:         C.Penalty2                 = 1;
188:         C.Penalty1[0]              = 0.0;
189:         C.Penalty1[1]              = 0.0;
190:         C.PresPenalty2 = 1;
191:         C.PresPenalty1[0]          = 0.0;
192:         C.PresPenalty1[1]          = 0.0;
193:         C.alpha                     = 1;
194:         C.StressPenalty = 1.0;
195:
196:
197:         //Gravity for full system
198:         C.GravityX_FormularObj = GetFormulaObject(InitialValues.
GravityXfunction);
199:         C.GravityY_FormularObj = GetFormulaObject(InitialValues.
GravityYfunction);
200:         C.GravityXX_FormularObj = GetFormulaObject(InitialValues.
GravityXXfunction);
201:         C.GravityXY_FormularObj = GetFormulaObject(InitialValues.
GravityXXfunction);
202:         C.GravityYY_FormularObj = GetFormulaObject(InitialValues.
GravityYYfunction);
203:
204:         //Set initial values
205:         C.InitialValues.Add("VelocityX", GetFormulaObject(InitialValues.
VelocityXfunction));
206:         C.InitialValues.Add("VelocityY", GetFormulaObject(InitialValues.
VelocityYfunction));
207:         C.InitialValues.Add("StressXX", GetFormulaObject(InitialValues.
StressXXfunction));
208:         C.InitialValues.Add("StressXY", GetFormulaObject(InitialValues.
StressXYfunction));
209:         C.InitialValues.Add("StressYY", GetFormulaObject(InitialValues.
StressYYfunction));
210:         C.InitialValues.Add("Phi", GetFormulaObject(InitialValues.Phi));
211:
212:
213:         C.AddBoundaryValue("Velocity_inlet", "VelocityX",
GetFormulaObject(InitialValues.VelocityXfunction));

```

```

214:         C.AddBoundaryValue("Velocity_inlet", "VelocityY",
    GetFormulaObject(InitialValues.VelocityYfunction));
215:         C.AddBoundaryValue("Velocity_inlet", "StressXX", GetFormulaObject
    (InitialValues.StressXXfunction));
216:         C.AddBoundaryValue("Velocity_inlet", "StressXY", GetFormulaObject
    (InitialValues.StressXYfunction));
217:         C.AddBoundaryValue("Velocity_inlet", "StressYY", GetFormulaObject
    (InitialValues.StressYYfunction));
218:         C.AddBoundaryValue("Velocity_inlet", "Pressure", GetFormulaObject
    (InitialValues.Pressurefunction));
219:         C.AddBoundaryValue("Velocity_inlet", "GravityX", GetFormulaObject
    (InitialValues.GravityXfunction));
220:         C.AddBoundaryValue("Velocity_inlet", "GravityY", GetFormulaObject
    (InitialValues.GravityYfunction));
221:
222:         //Save Session and next...
223:         C.SessionName = "LDG_NS_Convergence_Study_p"+degree+"_GridLevel"+
    elemInd;
224:         Controls.Add(C);
225:         elemInd += 1;
226:         Console.WriteLine("Created control: " + C.SessionName);
227:     }
228: }

```

```

229: // =====
230: // Launch Jobs
231: // =====

```

```

232: int[] procs = new int[] {1};
233: foreach(var ctrl in Controls) {
234:     foreach (int element in procs){
235:         var oneJob = ctrl.CreateJob();
236:         oneJob.NumberOfMPIProcs = element;
237:         oneJob.ExecutionTime = "24:00:00";
238:         oneJob.Activate(myBatch);
239:     }
240: }

```

A.2. Confined Cylinder Benchmark

```

241: restart

```

```

242: // using the Rheology Solver
243: using BoSSS.Application.Rheology;
244: using BoSSS.Solution.AdvancedSolvers;

```

```

245: // Workflow Manager Name for Postprocessing
246: WorkflowMgm.Init("ConfinedCylinder_p4_msh1_Wi=1.0_Re=0");

```

```

247: // Database
248: var myDb = OpenOrCreateDatabase(@"\\Path\to\Database\ConfinedCylinder");

```

```

249: // Batch for caculation on the HPC at FDY
250: //var myBatch = new MsHPC2012Client(@"\hpccluster\hpccluster-scratch\NAME\
    deploy_dir\", "hpccluster", ComputeNodes : new[] {"hpccluster"});
251:
252: //Batch for caculation on local PC
253: var myBatch = new MiniBatchProcessorClient(@"\Path\to\deploy\directory\
    ConfinedCylinder");

254: // Start MiniBatchProcessor
255: MiniBatchProcessor.Server.StartIfNotRunning();

256: // =====
257: // Convergence Parameters
258: // =====

259: // single grid and polynomial order
260: int degree = 4;
261: int gridNo = 1;
262:
263: //multiple grids and polynomial orders, e.g. for convergence study
264: //int[] pOrder = new int[] {1, 2, 3, 4};
265: //int numberGrids = 4;
266:
267: // #processors
268: int[] procs = new int[] {4};

269: // =====
270: // Physical Parameters
271: // =====

272: double u0          = 1.5; //max velocity at inlet
273: double h           = 4.0; // full channel height
274: double beta        = 0.59; // Newtonian to total viscosity
275: double Reynolds    = 1.0; // Reynolds number
276: double Weissenberg = 1.0; //aim Weissenberg number
277: bool RaiseWeissenberg = true;
278: double WeissenbergIncrement = 0.1; // increment for homothopy method
279: double giesekusfactor = 0.0; //mobility factor for Giesekus model
280: //((default: 0, means Oldroyd B)

281: // =====
282: // Timestepping: steady/unsteady
283: // =====

284: int NoOfTimesteps = 1; //5000;
285: double dt          = 1e6; //0.01;

286: // =====
287: // Init grids from gmsk and save to database
288: // =====

```

```

289: // loop for multiple grids
290: //GridCommons grids;      = new GridCommons[numberGrids];
291: //for (int k = 0; k < numberGrids; k++) {
292:
293: // define k for single grid
294: int k = gridNo;
295:
296: Gmsh gmshGrid = new Gmsh(@"\\...\Cylinder_GRIDS\Confined_Cylinder_mesh_"+k+"
    .msh");
297:
298: GridCommons bosssGrid = gmshGrid.GenerateBoSSSGrid();
299: bosssGrid.Name      = "confined_cylinder";
300:
301: bosssGrid.EdgeTagNames.Add(1, "Velocity_inlet");
302: bosssGrid.EdgeTagNames.Add(2, "Freeslip");
303: bosssGrid.EdgeTagNames.Add(3, "Wall_top");
304: bosssGrid.EdgeTagNames.Add(4, "Pressure_Outlet");
305: bosssGrid.EdgeTagNames.Add(5, "Wall_cylinder");
306:
307: Func<double[], byte> edgeTagFunc = delegate (double[] X) {
308:
309:     double x = X[0];
310:     double y = X[1];
311:
312:     if (Math.Abs(x - (-15)) < 1.0e-10)
313:         return 1;
314:     if (Math.Abs(x - (15)) < 1.0e-10)
315:         return 4;
316:     if (Math.Abs(y - (0)) < 1.0e-10)
317:         return 2;
318:     if (Math.Abs(y - (+2)) < 1.0e-10)
319:         return 3;
320:     if (0 < y && y < 1.0 && -1.0 < x && x < 1.0)
321:         return 5;
322:
323:     throw new ArgumentOutOfRangeException("at x = " + x + "and y = " + y);
324: };
325:
326: bosssGrid.DefineEdgeTags(edgeTagFunc);
327: myDb.SaveGrid(ref bosssGrid);
328:
329: // loop for multiple grids
330: //grids[k] = bosssGrid;
331: //}

```

```

332: // =====
333: // initial and boundary conditions
334: // =====

```

```

335: // Set Initial Conditions
336: static class InitialValues {
337:
338:     static double u0 = 1.5;
339:     static double h  = 4.0;
340:
341:     public static double VelocityXfunction(double[] X) {

```

```

342:         return u0 * (1 - (X[1] * X[1])/h);
343:     }
344:
345:     public static double VelocityYfunction(double[] X) {
346:         return 0.0;
347:     }
348:
349:     public static double Phi(double[] X) {
350:         return -1.0;
351:     }
352: }

```

```

353: //Set boundary conditions
354: var Wall = new Formula("X => 0");

```

```

355: // =====
356: // setup control object for a solver run
357: // =====

```

```

358: // for multiple control files, e.g. convergence study
359: //List<RheologyControl> Controls = new List<RheologyControl>();

```

```

360: //Controls.Clear();
361:
362: //loop for multiple polynomial orders
363: //foreach(int degree in pOrder) {
364: // int elemInd = 0;
365:
366: //loop for multiple grids
367: //foreach(GridCommons grd in grids) {
368:
369: // single grid
370: //GridCommons grd = bosssGrid;
371:
372:     //Database
373:     RheologyControl C = new RheologyControl();
374:     C.savetodb         = true;
375:     C.DbPath           = myDb.Path;
376:     C.ProjectName      = "Cylinder";
377:     C.SetGrid(grd);
378:     C.SetDGdegree(degree);
379:
380:     //in case of restart (initial values must be uncommented!)
381:     //Guid restartID = new Guid(">Session-ID<");
382:     //int ts         = -1; //last timestep
383:     //C.RestartInfo = new Tuple<Guid, TimestepNumber>(restartID, ts);
384:
385:     //SolverChooser
386:     C.NonLinearSolver.SolverCode          = NonLinearSolverCode.Newton;
387:     C.NonLinearSolver.MaxSolverIterations = 100;
388:     C.NonLinearSolver.MinSolverIterations = 3;
389:     C.NonLinearSolver.ConvergenceCriterion = 1E-6;
390:
391:     C.LinearSolver.MaxSolverIterations   = 100;
392:     C.LinearSolver.MinSolverIterations   = 3;
393:     C.LinearSolver.ConvergenceCriterion = 1E-6;

```

```

394: C.LinearSolver.SolverCode      = LinearSolverCode.classic_pardiso;
395:           // .classic_mumps; // .exp_Kcycle_schwarz_4Rheology;
396: C.LinearSolver.NoOfMultigridLevels = 1;
397: C.useFDJacobianForOperatorMatrix = false;
398:
399: //Timestepping
400: C.NoOfTimesteps      = NoOfTimesteps;
401: C.dt                 = dt;
402: C.dtMax              = C.dt;
403: C.dtMin              = C.dt;
404: C.Timestepper_Scheme = RheologyControl.TimesteppingScheme.BDF2;
405:           //ImplicitEuler;
406: C.ObjectiveParam     = 1.0;
407:
408: //Configuration Shock capturing and body forces
409: C.UsePerssonSensor   = false;
410: C.SensorLimit        = 1e-4;
411: C.AdaptiveMeshRefinement = false;
412: C.RefinementLevel    = 10;
413: C.UseArtificialDiffusion = false;
414: C.Bodyforces         = true;
415:
416: //Configuration IC and BC and Solver analytics
417: C.OperatorMatrixAnalysis = false;
418: C.SkipSolveAndEvaluateResidual = false;
419: C.SetInitialConditions   = true;
420: C.SetInitialPressure     = false;
421: C.SetParamsAnalyticalSol = false;
422: C.ComputeL2Error         = false;
423: C.Stokes                 = false; // no convection, linear system
424: C.StokesConvection       = true; //creeping flow, nonlinear
425: C.FixedStreamwisePeriodicBC = false;
426:
427: //Physical parameters
428: C.beta                   = beta;
429: C.Reynolds               = Reynolds;
430: C.Weissenberg            = Weissenberg;
431: C.RaiseWeissenberg       = RaiseWeissenberg;
432: C.WeissenbergIncrement   = WeissenbergIncrement;
433: C.giesekusfactor         = giesekusfactor;
434:
435: //Penalties
436: C.ViscousPenaltyScaling = 1;
437: C.Penalty2              = 1;
438: C.Penalty1[0]           = 0;
439: C.Penalty1[1]           = 0;
440: C.PresPenalty2          = 1.0;
441: C.PresPenalty1[0]       = 0.0;
442: C.PresPenalty1[1]       = 0.0;
443: C.alpha                 = 1;
444: C.StressPenalty         = 1.0;
445:
446: //Set initial values
447: C.InitialValues.Add("VelocityX", GetFormulaObject(InitialValues.
VelocityXfunction));
448: C.InitialValues.Add("VelocityY", GetFormulaObject(InitialValues.
VelocityYfunction));

```

```

449:     C.InitialValues.Add("Phi", GetFormulaObject(InitialValues.Phi));
450:
451:     //Set Boundary Conditions
452:     C.AddBoundaryValue("Wall_top", "VelocityX", Wall);
453:     C.AddBoundaryValue("Wall_top", "VelocityY", Wall);
454:     C.AddBoundaryValue("Wall_cylinder", "VelocityX", Wall);
455:     C.AddBoundaryValue("Wall_cylinder", "VelocityY", Wall);
456:
457:     C.AddBoundaryValue("Velocity_inlet", "VelocityX", GetFormulaObject(
        InitialValues.VelocityXfunction));
458:     C.AddBoundaryValue("Velocity_inlet", "VelocityY", GetFormulaObject(
        InitialValues.VelocityYfunction));
459:     C.AddBoundaryValue("Pressure_Outlet");
460:     C.AddBoundaryValue("FreeSlip");
461:
462:     //Save Session and next...
463:     C.SessionName = "ConfinedCylinder_p"+degree+"_meshNo"+elemInd+"_Wi="+
        Weissenberg+"_Re="+Reynolds+"_dt="+dt;
464:     Console.WriteLine("Created control: " + C.SessionName);
465:
466:     //end loops
467:     //Controls.Add(C);
468:     //elemInd += 1;
469: //}
470: //}

```

```

471: // =====
472: // Launch Jobs
473: // =====

```

```

474: //loop for multiple control files
475: //foreach(var ctrl in Controls) {
476:
477: //single control file
478: var ctrl = C;
479:     foreach (int element in procs){
480:         var oneJob          = ctrl.CreateJob();
481:         oneJob.NumberOfMPIProcs = element;
482:         oneJob.ExecutionTime   = "24:00:00";
483:         oneJob.Activate(myBatch);
484:     }
485: //}

```

A.3. Droplet in Shear Flow

```

486:     restart

```

```

487: // using the Rheology Solver
488: using BoSSS.Application.XRheology_Solver;;
489: using BoSSS.Solution.XNSECommon;
490: using BoSSS.Solution.Timestepping;
491: using BoSSS.Solution.XdgTimestepping;

```

```

492: using BoSSS.Solution.AdvancedSolvers;
493: using BoSSS.Foundation.XDG;
494: using BoSSS.Application.XNSE_Solver;

495: // Workflow Manager Name for Postprocessing
496: WorkflowMgm.Init("DropletInShearFlow_Chinyoka_AN_BN_Ca=60_m=1_Re=0.3_Wi=0.0
    _31Mar_1");

497: // Database
498: var myDb = OpenOrCreateDatabase(@"\\Path\to\Database\DropleInShearFlow");

499: // Batch for caculation on the HPC at FDY
500: //var myBatch = new MsHPC2012Client(@"\\hpccluster\hpccluster-scratch\NAME\
    deploy_dir\", "hpccluster", ComputeNodes : new[] {"hpccluster"});
501:
502: //Batch for caculation on local PC
503: var myBatch = new MiniBatchProcessorClient(@"\\Path\to\deploy\directory\
    DropleInShearFlow");

504: // Start MiniBatchProcessor
505: MiniBatchProcessor.Server.StartIfNotRunning();

506: // =====
507: // Convergence Parameters
508: // =====

509: int[] pOrder = new int[] {2};
510: int numberGrids = 1;
511: int [] numElements = new int[] {16};

512: // =====
513: // Create grid
514: // =====

515: int numE = numElements.Length;
516: double L = 2;
517: double H = 1;
518: GridCommons[] grids = new GridCommons[numElements.Length];
519: for (int k = 0; k < numE; k++) {
520:     double[] xNodes = GenericBlas.Linspace(0, L, 2 * numElements[k] + 1);
521:     double[] yNodes = GenericBlas.Linspace(0, H, numElements[k] + 1);
522:
523:     GridCommons grd;
524:     grd = Grid2D.Cartesian2DGrid(xNodes, yNodes, periodicX: false);
525:     grd.EdgeTagNames.Add(1, "Wall_lower");
526:     grd.EdgeTagNames.Add(2, "Wall_upper");
527:     grd.EdgeTagNames.Add(3, "pressure_outlet_1");
528:     grd.EdgeTagNames.Add(4, "pressure_outlet");
529:
530:     grd.DefineEdgeTags(delegate (double[] X) {
531:         byte et = 0;
532:         if(Math.Abs(X[1]) <= 1.0e-8)
533:             et = 1;
534:         if(Math.Abs(X[1] - H) <= 1.0e-8)

```

```

535:             et = 2;
536:         if (Math.Abs(X[0]) <= 1.0e-8)
537:             et = 3;
538:         if (Math.Abs(X[0] - L) <= 1.0e-8)
539:             et = 4;
540:         return et;
541:     });
542:
543:     myDb.SaveGrid(ref grd);
544:     grids[k] = grd;
545: }

```

```

546: // =====
547: // physical parameter
548: // =====

```

```

549: double reynoldsA = 0.3;
550: double reynoldsB = 0.3;
551: double betaA     = 0.5;
552: double betaB     = 0.5;
553: double sigma     = 4.3e-4;
554: double WeissenbergA = 0.0; // Newtonian Matrix
555: double WeissenbergB = 0.3; // Viscoelastic Droplet
556: double WeissenbergIncrement = 0.1;

```

```

557: // =====
558: // initial and boundary conditions
559: // =====

```

```

560: // Set Initial Conditions
561: static class InitialValues {
562:
563:     static double L = 2;
564:     static double H = 1;
565:     static double sigma = 4.3e-4;
566:     static double radius = 0.125;
567:
568:     public static double Phi(double[] X) {
569:         return ((X[0] - L/2).Pow2() + (X[1] - H/2).Pow2()).Sqrt() -
radius;
570:     }
571:
572:     public static double VelocityXfunction_A(double[] X) {
573:         return 2*X[1]-1;
574:     }
575:
576:     public static double VelocityXfunction_B(double[] X) {
577:         return 2*X[1]-1;
578:     }
579:
580:     public static double VelocityYfunction_A(double[] X) {
581:         return 0;
582:     }
583:
584:     public static double VelocityYfunction_B(double[] X) {
585:         return 0;

```

```

586:     }
587:
588:     public static double Pressurefunction_A(double[] X) {
589:         return sigma/radius;
590:     }
591:     public static double Wall_upper (double[] X) {
592:         return 1;
593:     }
594:     public static double Wall_lower (double[] X) {
595:         return -1;
596:     }
597:
598: }

```

```

599: var Wall_upper = new Formula("X => 1");
600: var Wall_lower = new Formula("X => -1");

```

```

601: // =====
602: // setup control object for a solver run
603: // =====

```

```

604: List<XRheology_Control> Controls = new List<XRheology_Control>();

```

```

605: Controls.Clear();
606: foreach(int p in pOrder) {
607:     int elemInd = 0;
608:     foreach(GridCommons grd in grids) {
609:
610:         var C = new XRheology_Control();
611:
612:         AppControl._TimesteppingMode TimesteppingMode = AppControl.
        _TimesteppingMode.Transient;
613:
614:         // grid and degree
615:         C.SetGrid(grd);
616:         C.SetDGdegree(p);
617:
618:         // basic database options
619:         C.DbPath = myDb.Path;
620:         C.savetodb = true;
621:         C.OperatorMatrixAnalysis = false;
622:         C.SkipSolveAndEvaluateResidual = false;
623:
624:         // Physical parameters
625:         C.PhysicalParameters.reynolds_A = reynoldsA;
626:         C.PhysicalParameters.reynolds_B = reynoldsB;
627:         C.PhysicalParameters.beta_a = betaA;
628:         C.PhysicalParameters.beta_b = betaB;
629:         C.PhysicalParameters.Sigma = sigma;
630:         C.PhysicalParameters.IncludeConvection = true;
631:         C.PhysicalParameters.Material = true;
632:
633:         C.RaiseWeissenberg = false;
634:         C.PhysicalParameters.Weissenberg_a = WeissenbergA;
635:         C.PhysicalParameters.Weissenberg_b = WeissenbergB;
636:         C.WeissenbergIncrement = WeissenbergIncrement;

```

```

637:
638:         // Initial values
639:         C.InitialValues.Add("Phi", GetFormulaObject(InitialValues.Phi));
640:         C.InitialValues.Add("VelocityX#A", GetFormulaObject(InitialValues
        .VelocityXfunction_A));
641:         C.InitialValues.Add("VelocityX#B", GetFormulaObject(InitialValues
        .VelocityXfunction_B));
642:         C.InitialValues.Add("Pressure#A", GetFormulaObject(InitialValues.
        Pressurefunction_A));
643:
644:         // boundary conditions
645:         C.AddBoundaryValue("Wall_upper", "VelocityX#A", Wall_upper);
646:         C.AddBoundaryValue("Wall_upper", "VelocityX#B", Wall_upper);
647:         C.AddBoundaryValue("Wall_lower", "VelocityX#A", Wall_lower);
648:         C.AddBoundaryValue("Wall_lower", "VelocityX#B", Wall_lower);
649:         C.AddBoundaryValue("pressure_outlet");
650:         C.AddBoundaryValue("pressure_outlet_1");
651:
652:         // misc solver options
653:         C.LinearSolver.SolverCode           = LinearSolverCode.
        classic_mumps;
654:         C.LinearSolver.NoOfMultigridLevels = 1;
655:         C.LinearSolver.MaxSolverIterations = 10;
656:         C.LinearSolver.ConvergenceCriterion = 1e-8;
657:
658:         C.NonLinearSolver.SolverCode       = NonLinearSolverCode.
        Newton;
659:         C.NonLinearSolver.MaxSolverIterations = 5;
660:         C.NonLinearSolver.MinSolverIterations = 1;
661:         C.NonLinearSolver.ConvergenceCriterion = 1e-8;
662:         C.NonLinearSolver.UsePresRefPoint   = false;
663:
664:         C.LevelSet_ConvergenceCriterion    = 1e-6;
665:         C.AdvancedDiscretizationOptions.ViscosityMode = ViscosityMode.
        Viscoelastic;
666:
667:         // level set options
668:         C.Option_LevelSetEvolution = (TimesteppingMode == AppControl.
        _TimesteppingMode.Steady) ? LevelSetEvolution.None : LevelSetEvolution.
        FastMarching;
669:
670:         C.LSContiProjectionMethod          = BoSSS.
        Solution.LevelSetTools.ContinuityProjectionOption.ConstrainedDG;
671:         C.AdvancedDiscretizationOptions.FilterConfiguration =
        CurvatureAlgorithms.FilterConfiguration.NoFilter;
672:         C.CutCellQuadratureType            =
        XQuadFactoryHelper.MomentFittingVariants.Saye;
673:
674:         // surface tension
675:         C.AdvancedDiscretizationOptions.SurfStressTensor =
        SurfaceSressTensor.FullBoussinesqScriven;
676:         C.PhysicalParameters.mu_I         = 1 * sigma;
677:         C.PhysicalParameters.lambda_I     = 2 * sigma;
678:
679:         C.AdvancedDiscretizationOptions.SST_isotropicMode =
        SurfaceStressTensor_IsotropicMode.LaplaceBeltrami_ContactLine;
680:

```

```

681:         C.AdvancedDiscretizationOptions.Penalty2 = 5;
682:         C.AdvancedDiscretizationOptions.Penalty1[0] = 0;
683:         C.AdvancedDiscretizationOptions.Penalty1[1] = 0;
684:         C.AdvancedDiscretizationOptions.UseWeightedAverages = false;
685:         C.InterAverage
           XBase_Control.InterfaceAveraging.weissenberg;
686:
687:         C.LogValues = XRheology_Control.LoggingValues.DropDeformation;
688:         C.LogPeriod = 1;
689:
690:         C.AdaptiveMeshRefinement = true;
691:         C.RefineStrategy
           CurvatureRefined;
           = XRheology_Control.RefinementStrategy.
692:         C.RefinementLevel
           = 3;
693:         C.BaseRefinementLevel
           = 3;
694:
695:         // timestepping
696:         switch (p) {
697:             case 1: {
698:                 C.Timestepper_Scheme = XRheology_Control.
TimesteppingScheme.ImplicitEuler;
699:                 C.Timestepper_BDFinit = TimeStepperInit.
SingleInit;
700:                 break;
701:             }
702:             case 2: {
703:                 C.Timestepper_Scheme = XRheology_Control.
TimesteppingScheme.ImplicitEuler;
704:                 C.Timestepper_BDFinit = TimeStepperInit.
SingleInit;
705:                 break;
706:             }
707:             default:
708:                 C.Timestepper_Scheme = XRheology_Control.
TimesteppingScheme.ImplicitEuler;
709:                 C.Timestepper_BDFinit = TimeStepperInit.
SingleInit;
710:                 break;
711:         }
712:
713:         C.Timestepper_LevelSetHandling = (TimesteppingMode == AppControl.
_TimesteppingMode.Steady) ? LevelSetHandling.None : LevelSetHandling.
LieSplitting;
714:
715:         C.TimesteppingMode = TimesteppingMode;
716:
717:         double dt
           = 1e-3;
718:         C.dtMax
           = dt;
719:         C.dtMin
           = dt;
720:         C.Endtime
           = 1000;
721:         C.NoOfTimesteps
           = 75000;
722:
723:         // misc
724:         C.saveperiod
           = 1;
725:         C.ComputeEnergy
           = false;
726:
727:         int kElem
           = numElements[elemInd];

```

```
728:         C.SessionName = "DropletInShear_Chinyoka";
729:
730:         Controls.Add(C);
731:         elemInd += 1;
732:         Console.WriteLine("Created control: " + C.SessionName);
733:
734:     }
735: }
```

```
736: // =====
737: // Launch Jobs
738: // =====
```

```
739: int[] procs = new int[] {1};
740: foreach(var ctrl in Controls) {
741:     foreach (int element in procs){
742:         var oneJob = ctrl.CreateJob();
743:         oneJob.NumberOfMPIProcs = element;
744:         oneJob.ExecutionTime = "24:00:00";
745:         oneJob.Activate(myBatch);
746:     }
747: }
```
