

Universidade do Minho
Escola de Engenharia

Bruno Manuel Oliveira Ramoa

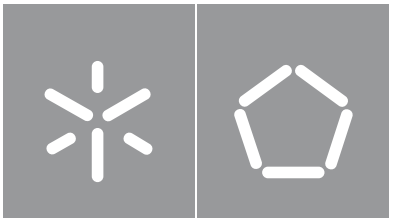
Computational Modeling of the Injection Molding Process: Open-Source Methods and Data-Driven Fiber Orientation

July 2026

Computational Modeling of the Injection Molding Process: Open-Source Methods and Data-Driven Fiber Orientation

Bruno Manuel Oliveira Ramoa

UMinho | 2026



Universidade do Minho
Escola de Engenharia

Bruno Manuel Oliveira Ramoa

Computational Modeling of the Injection Molding Process: Open-Source Methods and Data-Driven Fiber Orientation

Doctoral thesis in
Science and Engineering of Polymers and Composites

Work accomplished under the supervision of
Professor João Miguel de Amorim Novais da Costa Nóbrega
Professor Francisco Chinesta

July 2026

Direitos de autor e condições de utilização do trabalho por terceiros

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos. Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada. Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho:



Atribuição CC
BY
https://creativecommons.org/licenses/by/4.0/

Acknowledgements

I would also like to express my gratitude towards my supervisors: Professor João Miguel Nóbrega and Professor Francisco Chinesta, for their expertise, guidance, patience, availability and constructive feedback throughout the course of this project.

To my colleagues and friends for their encouragement and companionship.

Finally, I would like to thank my family. Their unwavering support and understanding was what made this work possible. This accomplishment is as much yours as it is mine.

The financial support provided by the Portuguese Foundation for Science and Technology under the PhD grant 2020.07424.BD and the projects UID/CTM/50025/2019 and UIDB/04436/2020.

Statement of integrity

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, 10th March 2026

Name: Bruno Manuel Oliveira Ramoa

Signature: _____

Abstract

In the automotive industry, the demand for lightweight components with excellent specific mechanical properties has driven the increasing adoption of fiber-reinforced thermoplastic materials (FRTM). However, manufacturing high-performance parts from this class of materials requires a detailed understanding of the material–process–properties interrelations across the injection molding (IM) cycle. Despite progress in IM modeling, there remains a clear need for transparent, reproducible and verifiable numerical tools that are easily extensible and adapt to application-specific requirements. Many of the existing solutions are proprietary (closed-source) and difficult to extend, often lacking systematic verification and reproducibility.

This project aimed to develop an open-source numerical toolbox to model the mold filling phase of the IM cycle. Building on OpenFOAM® 's modular architecture, the work enabled the implementation of a transparent and extensible framework designed to facilitate verification, reproducibility, and customization. The toolbox includes both compressible (commonly adopted in industrial practice) and incompressible (frequently used in academic studies) formulations, enabling consistent benchmarking and strengthening the link between research and industrial practice. This framework was complemented by modules for fiber-orientation modeling in FRTM, prioritizing robustness, traceability of model assumptions, and ease of validation.

Most fiber-orientation models available in the literature are phenomenological and often exhibit limited predictive capability for the fiber orientation evolution in complex IM parts. To address these limitations, the project integrated computational modeling and data science, leveraging machine learning techniques to construct data-driven models that predict fiber orientation. Two complementary approaches were pursued: (i) the inference of the closure relation required for FRTM modeling using a Neural ODE formulation, and (ii) the development of an orientation model based on rank-reduction autoencoders, enabling compact yet expressive representations of the orientation state while reducing computational burden.

Overall, this work contributed to the digital transformation of thermoplastics processing by delivering state-of-the-art open-source computational tools focused on transparency, robustness, and accuracy, and by leveraging the OpenFOAM® ecosystem to maximize impact.

Keywords: Computational modeling, Data-driven modeling, Fiber orientation, Injection molding, Polymer processing,

Resumo

Na indústria automóvel, a procura por componentes leves com excelentes propriedades mecânicas tem impulsionado a adoção de materiais termoplásticos reforçados com fibras (FRTM). Contudo, o fabrico de peças de alto desempenho com estes materiais exige uma compreensão detalhada das inter-relações material–processo–propriedades ao longo do ciclo de moldação por injeção (IM). Apesar de avanços na modelação do processo, continua a existir necessidade de ferramentas numéricas transparentes, reprodutíveis, verificáveis e facilmente adaptáveis. Muitas das soluções existentes são proprietárias (código fechado), difíceis de complementar ou não foram concebidas com verificação e reprodutibilidade como objetivos centrais, o que difficulta a sua utilização tanto na investigação académica como na promoção da inovação industrial.

Este projeto teve como objetivo desenvolver ferramentas numéricas de código aberto para modelar o ciclo de IM. Partindo da arquitetura modular do OpenFOAM®, o trabalho realizado permitiu a implementação de uma estrutura transparente e extensível, concebida para facilitar a verificação, a reprodutibilidade e a personalização. O conjunto de ferramentas suporta formulações compressíveis (comumente adotadas na prática industrial) e incompressíveis (frequentemente utilizadas em estudos académicos), permitindo *benchmarks* consistentes e reforçando a ligação entre a investigação e a prática industrial. O código numérico foi ainda complementado com módulos para previsão da evolução da orientação das fibras em FRTM, com especial atenção na robustez e facilidade de validação.

A maioria dos modelos de orientação de fibras disponíveis na literatura é de natureza fenomenológica e, frequentemente, demonstra capacidade limitada para prever a evolução da orientação das fibras em peças complexas produzidas por IM. Para ultrapassar estas limitações, o projeto integrou modelação computacional e ciência de dados, recorrendo a técnicas de inteligência artificial para construir modelos para a previsão da orientação das fibras. Foram seguidas duas abordagens complementares: (i) a inferência de uma relação de fecho necessária à modelação de FRTM, através de uma formulação utilizando *Neural ODE*, e (ii) o desenvolvimento de um modelo de orientação baseado em *autoencoders* com redução de ordem, permitindo representações compactas do estado de orientação das fibras.

De um modo geral, este trabalho contribuiu para a transformação digital da indústria de processamento de polímeros termoplásticos, ao desenvolver ferramentas computacionais de código aberto, assentes na transparência, robustez e precisão, tirando partido do ecossistema OpenFOAM® para maximizar a disseminação e impacto.

Palavras-chave: Modelação baseada em dados, Modelação computacional, Moldação por Injeção, Orientação de fibras, Processamento de polímeros,

Contents

Chapter 1 – Introduction 1

Chapter 2 – Fiber orientation modeling 5

2.1 Tensor based orientation models 8

2.2 Closures 12

Chapter 3 – Physics-Informed data-driven closure relation for dilute short fiber suspensions 19

Abstract 19

3.1 Introduction 20

3.2 Previous Data Driven Approaches 21

3.3 Data generation 23

3.4 Universal ordinary differential equations 26

3.4.1 Definition of bounds 27

3.4.2 Training 30

3.5 Cases and results 31

3.5.1 Homogeneous flows 31

3.5.2 Combined flow 32

3.5.3 Center-gated disk 34

3.5.4 Preliminary analysis in the presence of fiber interactions 36

3.6 Conclusion 37

Appendices 38

A Mandel Notation 38

B Parameterized bounds 38

C Availability of data and material 39

Chapter 4 – Non-Orthotropic Contributions in Flow-Induced Fiber Orientation: Toward Richer Closure Models 43

Abstract 43

4.1 Introduction 44

4.2 Data generation 45
4.3 Analysis of deviatoric components 46
4.3.1 Simple-shear 46
4.3.2 Biaxial flow with shear 48
4.3.3 Planar flow with shear 52
4.3.4 Uniaxial flow with shear 53
4.3.5 Other 57
4.4 Evaluation of orientation dynamics 60
4.5 Conclusion 63
Appendices 63
A Data generation with Stochastic approach 63
B Parametric bounds 64
C Availability of data and material 66
Chapter 5 – Rank Reduction Autoencoders applied to fiber orientation dynamics 70
Abstract 70
5.1 Introduction 71
5.2 Data generation 73
5.3 Training 75
5.4 Discussion 78
5.4.1 Latent space 78
5.4.2 Comparison with known closures 78
5.4.3 Prediction of ODF 80
5.4.4 Prediction of Stress 80
5.4.5 Prediction of second-order tensor 82
5.5 Predicting coefficient space 84
5.6 Conclusion 87
Appendix 87
Availability of data and material 87

Chapter 6 – A semi-automatic approach to assess the convergence order in OpenFOAM® 91
Abstract 91
6.1 Introduction 92
6.2 Methodology 93
6.2.1 Unsteady heat transfer problem 93
6.2.2 Method of manufactured solutions 94
6.2.3 Errors and convergence orders 95
6.3 Semi-automatic approach 99
6.4 Case studies 105
6.4.1 `laplacianFoam` test case studies 106
6.4.2 `simpleFoam` test case study 107
6.5 Results and Discussion 111
6.5.1 `laplacianFoam` test case studies 111
6.5.1.1 2D steady-state test case study 111
6.5.1.2 2D unsteady test case study 111
6.5.1.3 1D unsteady test case study 112
6.5.2 `simpleFoam` test case study 113
6.6 Conclusions 114
Appendix 115
Availability of data and material 115
Chapter 7 – An Open-Source Framework for Modeling the Evolution of Fiber Orientation 118
Abstract 118
7.1 Introduction 119
7.2 Modeling fiber orientation within the OpenFOAM ® framework 120
7.3 Closure model with symbolic computation 122
7.4 Case studies 125
7.4.1 Simple-shear case study 126
7.4.2 Center-gated disk case study 128

7.5 Results and discussion 129
7.5.1 Simple-shear case study 129
7.5.2 Center-gated disk case study 131
7.6 Conclusions 134
Appendix 134
Availability of data and material 134
Chapter 8 – A unifying open-source numerical framework for modeling injection molding of thermoplastic materials 138
Abstract 138
8.1 Introduction 140
8.2 Mathematical formulation 141
8.2.1 Filling 142
8.2.2 Packing 143
8.2.3 Cooling 143
8.2.4 Rheological models 144
8.2.5 Equation-of-state 144
8.2.6 Free surface modeling 145
8.2.7 Boundary conditions 146
8.2.8 Solution strategy 148
8.3 Solver verification 149
8.3.1 Method of manufactured solutions 149
8.3.1.1 Viscosity model 150
8.3.1.2 Equation-of-state 152
8.4 Phase advection 155
8.5 Analysis of dynamic outlet conditions 157
8.5.1 Cross-section of center-gated disk 157
8.5.2 Dogbone specimen 167
8.6 Solver validation 171
8.6.1 Spiral 172
8.6.2 Center gated disk 176

8.6.3 Fiber orientation modeling 181
8.7 Conclusion 185
Appendices 187
A Standard solution approach 187
B α-lagged solution approach 188
C Availability of data and material 188
Chapter 9 – Conclusion and Future work 196
9.1 General Conclusion 196
9.2 Future work 197

List of Figures

2.1 Unit vector $\mathbf{p}$ and (θ, ϕ) pair that define the orientation of a single fiber. 5

2.2 Illustration of fiber orientation states and the corresponding fiber orientation tensor, $\mathbf{A}$. . 8

3.1 Admissible eigenspace of orientation states in terms of eigenvalues. 22

3.2 Sampling of points over the surface of the unit sphere and the corresponding 2^{nd}-order orientation tensor. 24

3.3 Comparison of A_{11} component obtained through [25] (full line), and solving Jeffery's motion for a population of 10^4 fibers (dashed line). Simple-shear (SS), $v = \left[x_2,\ 0,\ 0\right]^{\mathrm{T}}$, planar extension (pl), $v = \left[x_1,\ -x_2,\ 0\right]^{\mathrm{T}}$, uniaxial (uni), $v = \left[-x_1/2,\ -x_2/2,\ x_3\right]^{\mathrm{T}}$, and biaxial (bi), $v = \left[x_1,\ x_2,\ -2x_3\right]^{\mathrm{T}}$ 26

3.4 Schematic representation of composite layer. 28

3.5 Admissible bounds of d_1, d_2 and d_3 for the isotropic state. 29

3.6 Average loss throughout each training epoch. 30

3.7 Model prediction for simple-shear(a), shear-stretch (b) and shear-planar(c). 33

3.8 Evolution of A_{11} and A_{22} for the combined flow test case. 34

3.9 Evolution of the main $\mathbf{A}_2$ tensor components along the normalized radius $(\bar{x}_1)$ at different $\bar{x}_3$ locations, for the center gated-disk case: (a) A_{11} and (b) A_{22}. 36

3.10 Comparison of the temporal evolution of A_{11} and A_{12} in simple-shear. 37

4.1 Trajectory of the different flows ($\mathbf{v}_1$ to $\mathbf{v}_5$) inside the UBT triangle for $C_I = 1 \times 10^{-2}$. 46

4.2 Temporal evolution of A_{ij} and d_i components under the homogeneous velocity field $\mathbf{v}_1$. Dash-dot line $\longrightarrow C_I = 0$, dashed line $\longrightarrow C_I = 1 \times 10^{-3}$, solid line $\longrightarrow C_I = 1 \times 10^{-2}$ 48

4.3 Split of active components of $\mathbb{A} : \mathbf{D}$ in Canonical space for flow $\mathbf{v}_1$ at $C_I = 1 \times 10^{-2}$. Dashed lines correspond to the relative contribution of each component. 49

4.4 Trajectory of principal directions of $\mathbf{A}$ for flow $\mathbf{v}_1$. The principal directions $\mathbf{e}_1^{\mathrm{A}}$, $\mathbf{e}_2^{\mathrm{A}}$, and $\mathbf{e}_3^{\mathrm{A}}$ are represented in red, blue, and green, respectively. The final position along each trajectory are marked with a circle, cross, and square for $C_I = 0$, $C_I = 1 \times 10^{-3}$, and $C_I = 1 \times 10^{-2}$, respectively. 49

4.5 Temporal evolution of A_{ij} and d_i components under the homogeneous velocity field $\mathbf{v}_2$. Dash-dot line $\longrightarrow C_I = 0$, dashed line $\longrightarrow C_I = 1\times10^{-3}$, solid line $\longrightarrow C_I = 1\times10^{-2}$.50

4.6 Split of active components of $\mathbb{A} : \mathbf{D}$ in Canonical space for flow $\mathbf{v}_2$ at $C_I = 1\times10^{-2}$. Dashed lines correspond to the relative contribution of each component. 51

4.7 Trajectory of principal directions of $\mathbf{A}$ for flow $\mathbf{v}_2$. The principal directions $\mathbf{e}_1^{\mathrm{A}}$, $\mathbf{e}_2^{\mathrm{A}}$, and $\mathbf{e}_3^{\mathrm{A}}$ are represented in red, blue, and green, respectively. The final position along each trajectory are marked with a circle, cross, and square for $C_I = 0$, $C_I = 1\times10^{-3}$, and $C_I = 1\times10^{-2}$, respectively. 52

4.8 Temporal evolution of A_{ij} and d_i components under the homogeneous velocity field $\mathbf{v}_3$. Dash-dot line $\longrightarrow C_I = 0$, dashed line $\longrightarrow C_I = 1\times10^{-3}$, solid line $\longrightarrow C_I = 1\times10^{-2}$.53

4.9 Trajectory of principal directions of $\mathbf{A}$ for flow $\mathbf{v}_3$. The principal directions $\mathbf{e}_1^{\mathrm{A}}$, $\mathbf{e}_2^{\mathrm{A}}$, and $\mathbf{e}_3^{\mathrm{A}}$ are represented in red, blue, and green, respectively. The final position along each trajectory are marked with a circle, cross, and square for $C_I = 0$, $C_I = 1\times10^{-3}$, and $C_I = 1\times10^{-2}$, respectively. 53

4.10 Temporal evolution of A_{ij} and d_i components under homogeneous velocity field $\mathbf{v}_4$. Dashed line $\longrightarrow C_I = 1\times10^{-3}$, solid line $\longrightarrow C_I = 1\times10^{-2}$. 54

4.11 Eigenvalues evolution for under the homogeneous flow $\mathbf{v}_4$ with $C_I = 1\times10^{-2}$ 55

4.12 Split of active components of $\mathbb{A} : \mathbf{D}$ in Canonical space for flow $\mathbf{v}_4$ at $C_I = 1\times10^{-2}$. 56

4.13 Trajectory of principal directions of $\mathbf{A}$ for flow $\mathbf{v}_4$. The principal directions $\mathbf{e}_1^{\mathrm{A}}$, $\mathbf{e}_2^{\mathrm{A}}$, and $\mathbf{e}_3^{\mathrm{A}}$ are represented in red, blue, and green, respectively. The final position along each trajectory are marked with a cross, and square for $C_I = 1\times10^{-3}$ and $C_I = 1\times10^{-2}$, respectively. 56

4.14 Temporal evolution of A_{ij} and d_i components under homogeneous velocity field $\mathbf{v}_4$. Dashed line $\longrightarrow C_I = 1\times10^{-3}$, solid line $\longrightarrow C_I = 1\times10^{-2}$. 57

4.15 Eigenvalues evolution for under the homogeneous flow $\mathbf{v}_5$ with $C_I = 1\times10^{-2}$ 57

4.16 Temporal evolution of A_{ij} and d_i components under homogeneous velocity field $\mathbf{v}_6$. Dashed line $\longrightarrow C_I = 1\times10^{-3}$, solid line $\longrightarrow C_I = 1\times10^{-2}$. 58

4.17 Split of active components of $\mathbb{A} : \mathbf{D}$ in Canonical space for flow $\mathbf{v}_6$ at $C_I = 1\times10^{-2}$. 59

4.18 Trajectory of principal directions of $\mathbf{A}$ for flow $\mathbf{v}_6$. The principal directions $\mathbf{e}_1^{\mathrm{A}}$, $\mathbf{e}_2^{\mathrm{A}}$, and $\mathbf{e}_3^{\mathrm{A}}$ are represented in red, blue, and green, respectively. The final position along each trajectory are marked with a cross and square for $C_I = 1\times10^{-3}$, and $C_I = 1\times10^{-2}$, respectively. 60

4.19 Prediction of d_i coefficients from a third-order polynomial model fitted with data in simple-shear with $C_I = 1\times10^{-2}$ 61

4.20 Prediction of the A_{ij} dynamics in simple-shear flow with $C_I = 1{\times}10^{-2}$ using different closure models. A_{ij}^{true} represents the true dynamics of the system, A_{ij}^{pred} model includes all d_i components, $A_{ij}^{\text{pred}^*}$ includes only the orthotropic trio and A_{ij}^{ORE} corresponds to the ORE closure 62

21 Comparison between ODF and stochastic approach in simple shear $\mathbf{v} = \begin{bmatrix} x_3, & 0, & 0 \end{bmatrix}^{\text{T}}$ with $C_I = 1 \times 10^{-2}$ 65

22 Admissibility region, in pink, for the pair (d_6, d_7) for simple-shear flow with $C_I = 1 \times 10^{-2}$ at 1 s and 25 s, respectively. Values are rounded to the 5^{th} decimal case . . . 66

5.1 RRAE with MLP network. k_{max} defines the number of modes retained from SVD decomposition. 76

5.2 Coefficients of the latent space. Highlight for samples with $\theta = 0\,[rad]$. 79

5.3 Representative curves of ODF. True data is represented with a full line, predicted data with a dashed line. 80

5.4 Error distribution of ψ over the parametric space. 81

5.5 Prediction of $\boldsymbol{\tau}$ 81

5.6 Relative error distribution of $\boldsymbol{\tau}$. 82

5.7 Relative error of $\boldsymbol{\tau}$ for each closure model. Data bellow average has been omitted. . . . 83

5.8 Prediction of $\mathbf{A}_2$ 83

5.9 Relative errors of $\mathbf{A}_2$ for each closure model. Data bellow average has been omitted. . . 84

5.10 Relative errors of $\mathbf{A}_2$ over the parametric space 84

5.11 Coefficient space surrogate. Basis is already computed during the training process. . . . 85

5.12 Coefficient space relative error. Values bellow average have been omitted. 86

6.1 Flow chart of the semi-automatic approach based on the MMS in OpenFOAM®. 99

6.2 2D domain with representation of the corresponding patches (dimensions in meters). . . 105

6.3 Manufactured solutions for the verification of the `laplacianFoam` solver. 106

6.4 Manufactured solutions for the verification of the `simpleFoam` solver. 109

7.1 Concept for the `functionObject` for modeling fiber orientation. 121

7.2 Geometry, mesh and patches for the simple-shear case study. 127

7.3 Part and mesh representation with the corresponding patches for the center-gated disk case study (dimensions in m). 128

7.4 Mesh refinement study with the Folgar-Tucker model and the Hybrid closure for the simple-shear case study. . . . 130

7.5 Mesh refinement study with the Folgar-Tucker model and the IBOF closure for the simple-shear case study. . . . 131

7.6 Mesh refinement study with the Folgar-Tucker model and the ORE closure for the simple-shear case study. . . . 131

7.7 Mesh refinement study with the iARD and MRD models for the A_{xx} component for the simple-shear case study . . . 132

7.8 Map of shear/stretch ratio for the center gated disk. . . . 132

7.9 Mesh refinement study with the Folgar-Tucker model and the IBOF closure for the center-gated disk case study. . . . 133

8.1 Solution field for u, v, p and resulting viscosity μ in Taylor-Green test case . . . 152

8.2 Solution field for u, v, T and p for the Veeraragavan et al. [81] solution . . . 154

8.3 Comparison of algebraic and geometric VoF algorithms in vortical flow. . . . 156

8.4 Geometry and boundary definition for specimens used in the dynamic outlet analysis (dimensions in mm) . . . 157

8.5 Comparison of VoF field for the cross-sectional disk as a function of time. Sequence was obtained by clipping the VoF field with $\alpha \geq 0.5$ . . . 160

8.6 Comparison of pressure history at the inlet between standard approach, $(\cdot)_{std}$, and α-lagged strategy, $(\cdot)_{fix}$ . . . 161

8.7 VoF balance of CGD cross-section test case . . . 163

8.8 Comparison of VoF field for the cross-sectional disk as a function of time. Sequence was obtained by clipping the VoF field with $\alpha \geq 0.9$ . . . 164

8.9 Velocity profile at $x = 28$ mm at 0.09 s . . . 164

8.10 Comparison of VoF field for the CGD cross-section as a function of time. Conditions BC2-BC4 have null-flux through the dynamic outlets. The image sequence was obtained by clipping the VoF field with $\alpha \geq 0.9$ . . . 166

8.11 Pressure history and velocity profile of the analyses conditions. Cases BC2-BC4 used a no α-flux boundary condition . . . 166

8.12 Comparison of the polymer region evolution for the dogbone case study. The sequence was obtained by clipping the VoF field with $\alpha \geq 0.5$ . . . 168

8.13 VoF balance of dogbone test case . . . 169

8.14 Pressure history for BC1-BC4 using the null-normal-gradient boundary conditions 170

8.15 Comparison of VoF field for the dogbone specimen as a function of time. Sequence was obtained by clipping the VoF field with $\alpha \geq 0.9$. No-flux boundary conditions were used for BC2-BC4 170

8.16 Pressure history for BC1-BC3 using the no-flux boundary conditions 171

8.17 Geometry and boundary patches for the spiral case study. Dimensions in millimeters . . 172

8.18 Computational meshes employed for the spiral test case. 173

8.19 Cavity filling sequence for Moldex3D®. Each image represents the fill time at 0.25 s intervals, with the exception of the last one 174

8.20 Cavity filling sequence for `IMFoam`. Each image represents the fill time at 0.25 s intervals, with the exception of the last. Colormap shows regions where $\alpha \geq 0.9$ 175

8.21 Comparison of results between applications for the spiral test case. 175

8.22 Geometry and boundary patches for the CGD case study 176

8.23 Computational domain for the CGD test case. 176

8.24 Fill sequence of Moldex3D®. Each image is represent the fill time at 0.1 s intervals. . . 179

8.25 Fill sequence of `IMFoam`. Each image is represent the fill time at 0.1 s intervals. Colormap shows regions where $\alpha \geq 0.9$ 180

8.26 Pressure field at the $t = 1$ s 180

8.27 Comparison of the computed velocity profile at three distinct locations along the radius of the CGD (left), and pressure along the radius (right) at $t = 1$ s. Symbols are data collected from Moldex3D®, lines are from OpenFOAM® 181

8.28 Comparison of the the pressure history between Moldex3D® and OpenFOAM® 181

8.29 Geometry and boundary patches of the FGD study. Dimensions in millimeters 182

8.30 Computational domain for the fan gate case 182

8.31 Fill sequence of Moldex3D®. Each image is represent the fill time at 0.1 s intervals. . . 184

8.32 Fill sequence of `IMFoam`. Each image is represent the fill time at 0.1 s intervals. 184

8.33 Comparison of results between applications for the FGD test case 185

8.34 Comparison of orientation results between applications for the FGD test case at 99% fill. 185

35 Flow chart of solution algorithm with standard approach 187

36 Flow chart of solution algorithm with α-lagged approach 188

List of Tables

3.1 Flows used for training. G is a scaling parameter. 26

3.2 Error metrics for homogeneous flows 32

3.3 Error metrics calculated along the normalized disk thickness ($\bar{x}_3$) for the center-gated disk case 35

4.1 Resume of observed transitions in non-orthotropic components 60

5.1 Parameters used to generate ODF at steady-state 74

5.2 Network structure 77

5.3 Errors in dataset for different models 79

5.4 Relative errors in the prediction of coefficients of latent space 86

5.5 Relative errors in dataset using predicted coefficients 86

6.1 Temporal and spatial discretization schemes for the verification of the `laplacianFoam` solver (`fvSchemes` dictionary). 107

6.2 Solution methods for the verification of the `laplacianFoam` solver (`fvSolution` dictionary). 107

6.3 Temporal and spatial discretization schemes for the verification of the `simpleFoam` solver (`fvSchemes` dictionary). 109

6.4 Solution methods for the verification of the `simpleFoam` solver (`fvSolution` dictionary). 110

6.5 Parameters for the SIMPLEC calculation procedure of the `simpleFoam` solver (`SIMPLE` sub-dictionary from `fvSolution` dictionary). 111

6.6 Error norms obtained in the 2D steady-state test case study for the `laplacianFoam` solver. 111

6.7 Convergence orders obtained in the 2D steady-state test case study for the `laplacianFoam` solver. 112

6.8 Error norms obtained in the 2D unsteady test case study for the `laplacianFoam` solver. 112

6.9 Convergence orders obtained in the 2D unsteady test case study for the `laplacianFoam` solver. 113

6.10 Errors obtained in the 1D unsteady test case study for the `laplacianFoam` solver. . . 113

6.11 Convergence orders obtained in the 1D unsteady test case study for the `laplacianFoam` solver. . . . 113

6.12 Error norms obtained for u, v, and p in the 2D steady-state test case study for the `simpleFoam` solver. . . . 114

6.13 Convergence orders obtained for u, v, and p in the 2D steady-state test case study for the `simpleFoam` solver. . . . 114

7.1 Fiber orientation model parameters for the simple-shear case study. . . . 127

7.2 Temporal and spatial discretization schemes for the simple-shear case study. . . . 127

7.3 Solution methods for the simple-shear case study. . . . 128

7.4 Fiber orientation model parameters for the center-gated disk case study. . . . 129

8.1 Parameters of Cross-WLF model used in Taylor-Green test case. . . . 151

8.2 Finite volume discretization schemes for the Taylor–Green test case. . . . 151

8.3 Convergence orders obtained for the different error norm and variables u, v, and p for the Taylor-Green test case with Cross-WLF rheological model. . . . 152

8.4 Parameters of the modified-Tait model. All omited parameters are set to zero. . . . 154

8.5 Convergence orders obtained for the different error norms and variables u, v, p and T test case with modified-Tait model. . . . 154

8.6 Combination of boundary conditions used in the CGD cross-section tests. The acronyms *zG*, *fV*, *dO*, and *ns* denote: null normal gradient, fixed value, dynamic outlet, and no-slip, respectively. For the dynamic outlets, the bracketed label indicates the weighting function used (jump, sigmoidal, or linear) . . . 158

8.7 Discretization schemes for the CGD cross-section. . . . 159

8.8 Solution settings for the CGD cross-section. . . . 159

8.9 Modified solution settings for the dogbone test case. . . . 167

8.10 Discretization schemes for the spiral test case. . . . 173

8.11 Discretization schemes for the CGD. . . . 177

8.12 Boundary conditions used in three dimensional CGD case. The acronym zG stands for `zeroGradient`, fV for `fixedValue`, dO for `dynamicOulet`, CHT for `convectiveHeatFlux` . . . 178

8.13 Rheological and Equation-of-state parameters for PP Moplen HP 500N . . . 178

Nomenclature

Abbreviations

ARD	Anisotropic rotary diffusion
BICGStab	Biconjugate Gradients Stabilized Method
CGD	Center gated disk
CSE	Common sub-expression elimination
DIC	Diagonal-based incomplete Cholesky
DILU	Diagonal-based Incomplete LU preconditioner
FGD	Fan-gated disk
FRTM	Fiber reinforced thermoplasic materials
FT	Folgar-Tucker
FVM	Finite Volume Method
GAMG	Geometric algebraic multigrid
iARD	Improved anisotropic rotary diffusion
IBOF	Invariant-based optimal fitting
IM	Injection molding
IRD	Isotropic rotary diffusion
ML	Machine learning
MLP	Multi-layer perceptron
MMS	Method of manufactured solutions
MPI	Message Passing Interface
MRD	Moldflow rotational diffusion
MS	Manufactured solution
MULES	Multi-dimensional Universal Limiter for Explicit Solutions

ODEs Ordinary differential equations

ODF Orientation distribution function

ORE Orthotropic equivalent of the natural closure

pARD Principal anisotropic rotary diffusion

PCG Preconditioned conjugate gradient

PDE Partial differential equation

PISO Pressure-Implicit Split Operator

ReLU Rectified linear unit

RMS Root mean squared

RPR Retarding principal rate

RRAE Rank Reduction Autoencoder

RSC Reduced strain closure

SIMPLEC SIMPLE-Consistent

SIMPLE Semi-Implicit Method for Pressure-Linked Equations

SVD Singular value decomposition

TG Taylor-Green

UODEs Universal ordinary differential equations

VoF Volume-of-Fluid

Chapter 1 Introduction

Plastic materials are fully tied to our daily lives. Products made of plastic materials cover a wide range of applications, from domestic appliances to highly technical and sophisticated scientific products. In Europe, the plastics industry is responsible for over 1.5 million jobs, with a turnover of approximately 398 billion euros in 2025 [1].

Plastic products can be manufactured through different processing techniques, e.g., extrusion, rotational molding, thermoforming, blow molding, calendering, additive manufacturing, among others. In particular, injection molding (IM) allows great flexibility to obtain complex geometries, as well as high production rates with tight dimensional tolerances. The manufacturing technique consists of a cyclic process in which the molten (plasticized) plastic, conveyed from the plasticizing unit, is forced (injected) into a mold cavity with the geometry of the final part, where it is held under pressure until it cools down and solidifies. The mold can comprise a single or multiple cavities of similar or dissimilar shapes. The processing cycle ends with the opening of the mold and the subsequent extraction of the part(s).

Although the description of the IM process seems straightforward, the underlying thermo-mechanical environment is very complex. The process is transient, involves non-linear complex rheological materials, and exhibits high temperature gradients and ranges associated with the flow and cooling mechanisms. The complexity of the parts is arbitrary, and can include non-uniform wall thicknesses, ribs, bosses, undercuts, etc. A thorough description of the IM process and the overall processing cycle, process modeling information, and the most common problems (e.g., shrinkage, warpage and deformations) and defects associated with IM can be found in [2–4].

Given the above framework and the high costs associated with the manufacturing of molds for the IM process, several efforts have been made to numerically model the process, aiming at preventing design and technological errors at the design stage, in order to minimize cost losses and shortcomings at the manufacturing and production stages, leading to better parts and faster time-to-market [5].

Due to the ongoing demand for better and improved product performance, it is common to resort to fiber reinforced thermoplasic materials (FRTM) to achieve specifications unattainable with conventional plastics. For instance, in the automotive industry, many demanding parts are made of FRTM, because they enhance fuel efficiency through their exceptional specific mechanical properties [6]. However, the behavior of parts made of FRTM and manufactured by IM is highly dependent on the processing parameters and the thermo-mechanical history, fiber orientation, fiber length and concentration, among other factors whose influence is complex to establish.

Understanding fiber orientation in FRTM parts can substantially enhance the analysis of shrinkage, warpage, and structural behaviors [7–9]. Thus, computational modeling tools must be able to handle FRTM properly and accurately, within a time frame compatible with industry needs, to provide useful support in designing tasks, where a deep understanding of the process-structure-properties interrelations is required [10–12].

Despite the existence and maturity of commercial packages for injection molding process, significant limitations remain in both research and industrial transfer. From a computational research standpoint, many developments are made in in-house applications, making the independent verification of the numerical methods and reproducibility of results difficult.

From the material modeling perspective, predicting fiber orientation in FRTM remains an open challenge, with orientation models using a tensor-based formulations for computational efficiency. However, the underlying models are phenomenological in nature and often require a case-by-case parameter calibration. Additionally, the evolution equation of the tensor-based models involves the next high-order orientation tensor and, therefore, requiring a closure formulation to solve the system.

Motivated by the above limitations, the goals of this work are twofold: (i) the development of an open-source toolbox to model the most relevant phases of the IM cycle using state-of-the-art numerical methods, supporting both compressible formulations (commonly required for industrial practice) and incompressible formulations (often considered in academic studies); and (ii) the development of novel data-driven approaches to model fiber orientation in FRTM using artificial intelligence techniques.

This thesis is organized in 9 chapters, whose content resulted from the work done in the associated PhD project, and were adapted from manuscripts submitted or accepted for publication in relevant international peer-reviewed journals.

Regarding IM, the main contributions of this work include the development of an open-source computational framework, focusing on reproducibility, numerical verification, and extensibility. This framework addresses both incompressible and compressible formulations of thermoplastics and FRTM, including the compilation and improvement of state-of-the-art methodologies for phase advection, boundary conditions, and solution algorithms, considering the characteristics of the IM process.

With code verification and validation in mind, a computational framework has been developed to mitigate the challenges associated with the application of the Method of Manufactured Solutions (MMS). By streamlining the generation of manufactured solutions, source terms, and associated boundary conditions, the proposed approach offers a semi-automatic framework for code verification while preserving flexibility and generality across various physical models and numerical discretizations. Consequently, it contributes to more robust and reliable verification practices in computational modeling.

On the fiber orientation modeling front, this work proposes a new methodology to infer a closure relation for solving the evolution equation of fiber orientation models using a Neural ODE

formulation, and defines physically admissible bounds for fourth-order orientation tensors under both orthotropic and non-orthotropic symmetries. It also introduces a data-driven fiber orientation modeling framework based on rank-reduction autoencoders, that uses compressed information from rich orientation descriptors and suspension rheology.

These contributions aim to lower the research barrier concerning the IM process and orientation modeling while enriching the understanding of material-process-properties interrelationships, supporting the digital transformation of polymer processing through transparent and robust computational tools.

Each chapter of this thesis contains its own state-of-the-art, results, discussion, and conclusions. The original introductions from the associated papers were revised to eliminate redundancy. Additionally, a summarized version of the background on fiber orientation modeling is presented in the next section, to avoid repetition.

References

[1] Plastics europe: Plastics - the fast facts 2025. . Accessed: 2026-01-10.

[2] Huamin Zhou. *Computer Modeling for Injection Molding: Simulation, Optimization, and Control*. John Wiley and Sons, Inc., 2013.

[3] Peter Kennedy and Rong Zheng. *Flow Analysis of Injection Molds*. Carl Hanser Verlag GmbH and Co. KG, 2nd edition edition, 2013.

[4] Randy Kerkstra and Steve Brammer. Injection molding advanced troubleshooting guide. In Randy Kerkstra and Steve Brammer, editors, *Injection Molding Advanced Troubleshooting Guide (Second Edition)*, pages I–XX. Hanser, second edition edition, 2021.

[5] Tomasz Jachowicz, Ivan Gajdoš, Vlastimil Cech, and Volodymyr Krasinskyi. The use of numerical analysis of the injection process to select the material for the injection molding. *Open Engineering*, 11(1):963–976, 2021.

[6] Chao-Tsai Huang and Huan-Chang Tseng. Simulation Prediction of the Fiber Breakage History in Regular and Barrier Structure Screws in Injection Molding. *Polymer Engineering and Science*, 58(4):452–459, 2017.

[7] Huan-Chang Tseng, Chia-Hsiang Hsu, and Rong-Yeu Chang. Long fiber orientation and structural analysis using moldex3d, digimat and abaqus simulations. In *SPE ANTEC Conference*, 05 2017.

[8] Jonathan Köbler, Matti Schneider, Felix Ospald, Heiko Andrä, and Ralf Müller. Fiber orientation interpolation for the multiscale analysis of short fiber reinforced composite parts. *Computational Mechanics*, 61(6):729–750, 2018.

[9] Cathelin, Julien. Through-process modelling for accurate prediction of long term anisotropic mechanics in fibre reinforced thermoplastics. *MATEC Web Conf.*, 165:17002, 2018.

[10] Rong Zheng, Roger I. Tanner, and Xi-Jun Fan. *Injection Molding: Integration of Theory and Modeling Methods*. Springer-Verlag Berlin Heidelberg, 2011.

[11] Célio Fernandes, António José Pontes, Júlio César Viana, and António Gaspar-Cunha. Modeling and Optimization of the Injection-Molding Process: A Review. *Advances in Polymer Technology*, 37(2):429–449, 2018.

[12] J. Azaiez, K. Chiba, F. Chinesta, and A. Poitou. State-of-the-art on numerical simulation of fiber-reinforced thermoplastic forming processes. *Archives of Computational Methods in Engineering*, 9(2):141–198, 2002.

Chapter 2 Fiber orientation modeling

The employed formulation considers rigid, axi-symmetric, straight fibers, such as glass fibers typically used to manufacture FRTM [1], whose orientation characterization can be carried out at different scales.

At the micro-scale, the orientation of individual fibers is typically described by a unit vector $\mathbf{p}$ that points along the fiber axis or by a pair of (θ, ϕ) angles, as depicted in Figure 2.1. The Cartesian components of the unit vector are written as a function of (θ, ϕ) as: $\mathbf{p} = \left[\sin\theta\cos\phi,\ \sin\theta\sin\phi,\ \cos\theta\right]^{\mathrm{T}}$.

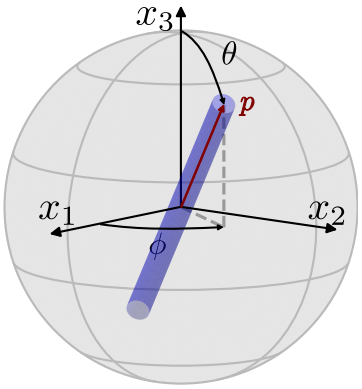


Figure 2.1: Unit vector $\mathbf{p}$ and (θ, ϕ) pair that define the orientation of a single fiber.

The hydrodynamic motion of a single ellipsoidal particle suspended in a Newtonian fluid was described in the seminal work of Jeffery [2], as:

$$\dot{p}_i = W_{ij}p_j + \xi\left(D_{ij}p_j - \left(D_{jk}p_kp_j\right)p_i\right), \tag{2.1}$$

where:

- $\dot{p}_i$ is hydrodynamic time rate-of-change of p_i,
- $\mathbf{W}$ is the vorticity tensor, given by: $\mathbf{W} = \frac{1}{2}\left(\mathbf{L} - \mathbf{L}^{\mathrm{T}}\right)$,
- $\mathbf{D}$ is the rate-of-deformation tensor, given by: $\mathbf{D} = \frac{1}{2}\left(\mathbf{L} + \mathbf{L}^{\mathrm{T}}\right)$,

- $\mathbf{L}$ is the velocity gradient tensor, given by $L_{ij} = \frac{\partial v_i}{\partial x_j}$,
- v_i are the components of the velocity vector,
- ξ is the particle shape factor, given by $\xi = \frac{a_r^2 - 1}{a_r^2 + 1}$, and
- a_r is the fiber aspect ratio.

At a meso-scale, the individuality of a fiber is replaced by a statistical approach, using a symmetric probability density function (ψ), also known as orientation distribution function (ODF) , describing the probability of a fiber lying between $\mathbf{p}$ and $\mathbf{p} + \mathrm{d}\mathbf{p}$ [3, 4]. As a descriptor of the orientation of a fiber ensemble, the ODF must satisfy several physical requirements. The periodicity requirement states that fibers have no preferred end, thus there are two possible choices for $\mathbf{p}$, and the resulting ODF must be periodic, i.e. $\psi(\mathbf{p}) = \psi(-\mathbf{p})$ [4]. The non-negative requirement states that $\psi(\theta, \phi) \geq 0, \ \forall\, (\theta, \phi) \in \mathcal{S}^2$ [5]. The normalization requirement states that, since every fiber must have some direction, the integral over the conformational space must be unitary [4], as follows:

$$\int_{\phi=0}^{\phi=2\pi} \int_{\theta=0}^{\theta=\pi} \psi(\theta, \phi) \sin\theta \, \mathrm{d}\theta \mathrm{d}\phi = 1 \tag{2.2}$$

Finally, the continuity requirement, where ψ is treated as a convected quantity and a conservation equation is derived, reading:

$$\frac{\mathrm{D}\psi}{\mathrm{D}t} = -\nabla_S \cdot \mathbf{q}^{\mathrm{J}}, \tag{2.3}$$

where $\mathrm{D}/\mathrm{D}t$ is the material derivative, ∇_S is the gradient operator on the surface of the unit sphere and $\mathbf{q}^{\mathrm{J}}$ is the probability flux due to Jeffery motion, given by $\mathbf{q}^{\mathrm{J}} = \psi\dot{\mathbf{p}}$ [6].

To capture the effect of fiber-fiber interactions, the traditional modeling approach in the literature is to incorporate a phenomenological diffusion term [1, 3, 7]. When introduced into the continuity equation of ψ, Equation (2.3), yields:

$$\frac{\mathrm{D}\psi}{\mathrm{D}t} = -\nabla_S \cdot \left(\mathbf{q}^{\mathrm{J}} + \mathbf{q}^{\mathrm{D}}\right) \tag{2.4a}$$

$$\mathbf{q}^{\mathrm{D}} = -\dot{\gamma}\mathbf{D}^{\mathrm{r}} \cdot \nabla_S \psi \tag{2.4b}$$

$$\mathbf{D}^{\mathrm{r}} = \mathbf{T}^{\mathrm{T}} \cdot \mathbf{C} \cdot \mathbf{T} \tag{2.4c}$$

where $\mathbf{T}$ is a 3×2 projection matrix from Cartesian space to a surface tensor on the unit sphere, given by Equation (2.5) [1, 6], $\mathbf{C}$ is the rotary diffusion tensor, and $\mathbf{D}^{\mathrm{r}}$ is the surface rotary diffusion tensor.

$$\mathrm{T} = \begin{bmatrix} \cos\theta\cos\phi & -\sin\phi \\ \cos\theta\sin\phi & \cos\theta \\ -\sin\theta & 0 \end{bmatrix} \tag{2.5}$$

Equation (2.4b) shows a general anisotropic form of the diffusive flux. The scalar magnitude of the symmetric part of the velocity gradient, $\dot{\gamma}$, stems from the seminal work of Folgar and Tucker [3], in which it was observed that fibers only re-orient if the suspension is deforming [7]. To capture this phenomenon, the diffusion flux was made proportional to $\dot{\gamma}$.

Although the ODF is a complete orientation descriptor, for practical usage in numerical tools to predict the flow-induced orientation, the solution of Equation (2.4) can be computationally prohibitive due to its high dimensionallity, requiring discretization over time, space, and conformational space [8]. Accordingly, to condense the information contained in the ODF, one suitable approach is the usage of macro-descriptors known as orientation/moment tensors [4]. These are defined from the statistical moments of the ODF as :

$$\mathbf{A}^{\mathrm{k}} = \int_S \psi(\mathbf{p})\ \mathbf{p}^{\otimes \mathrm{k}}\ \mathrm{d}\mathbf{p}, \tag{2.6}$$

with $\mathbf{p}^{\otimes \mathrm{k}}$ being the k^{th} dyadic product of $\mathbf{p}$ with itself [5]. The most commonly used being the second ($\mathbf{A}$), and fourth-order ($\mathbb{A}$) tensors. These descriptors are symmetric with respect to any pair of indices and positive-semi-definite. Due to the normalization condition, the trace of $\mathbf{A}$ is unitary, and the information of the lower order tensor is contained in the next even-ordered tensor, i.e. $A_{ijkk} = A_{ij}$ [4–6].

The resulting evolution equation for the second-moment tensor, using an anisotropic rotary diffusion, reads [1, 7]:

$$\dot{\mathbf{A}} = \dot{\mathbf{A}}^{\mathrm{J}} + \dot{\mathbf{A}}^{\mathrm{D}}, \tag{2.7a}$$

$$\dot{\mathbf{A}}^{\mathrm{J}} = \mathbf{W}\cdot\mathbf{A} - \mathbf{A}\cdot\mathbf{W} + \xi\left(\mathbf{D}\cdot\mathbf{A} + \mathbf{A}\cdot\mathbf{D} - 2\mathbb{A}:\mathbf{D}\right) \tag{2.7b}$$

$$\dot{\mathbf{A}}^{\mathrm{D}} = \dot{\gamma}\left[2\mathbf{C} - 2\mathrm{tr}\left(\mathbf{C}\right)\ \mathbf{A} - 5\left(\mathbf{C}\cdot\mathbf{A} + \mathbf{A}\cdot\mathbf{C}\right) + 10\mathbb{A}:\mathbf{C}\right], \tag{2.7c}$$

Although computationally more efficient than the ODF, the use of orientation tensors have some drawbacks:

- The reconstruction of the ODF from $\mathbf{A}$ is ambiguous, as $\mathbf{A}$ might not represent a unique orientation state [9]. Figure 2.2 illustrates different fiber configurations that result in the same orientation tensor.

- In the derivation of the transport equation for the orientation tensor, the next even-order tensor appears. To close the system, a closure relationship is needed, and several authors have proposed solutions to write $A_{ijkl} = f(A_{ij})$. In Section 2.2, the most relevant closure relations used in literature will be presented.

Nevertheless, this descriptor retains the generality of the information in the ODF and provides a compact, computationally efficient and convenient representation of the fiber orientation state, being the state-of-the-art descriptor used in CAE applications.

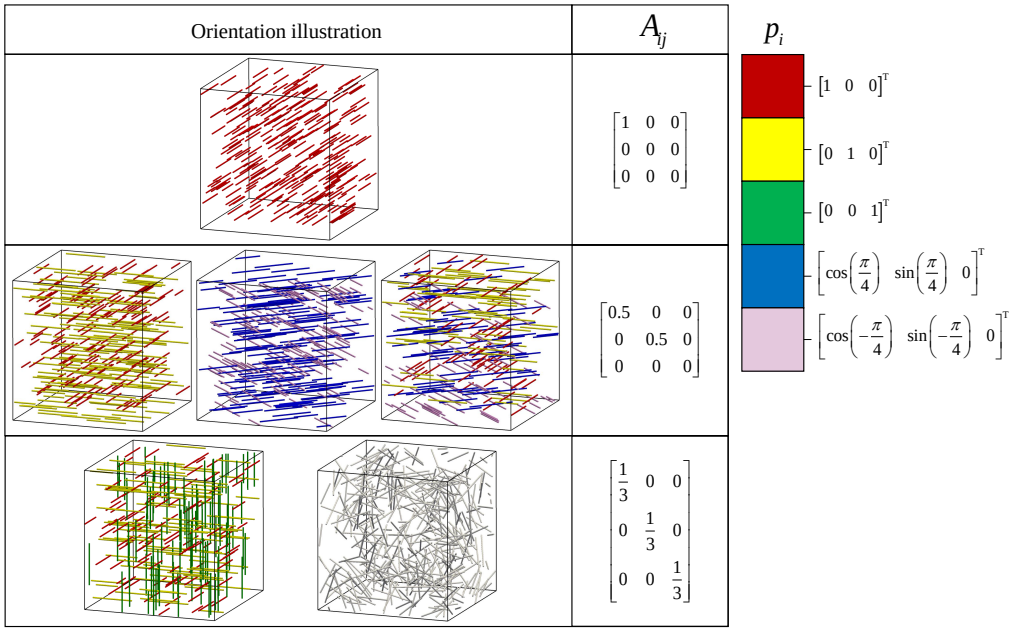


Figure 2.2: Illustration of fiber orientation states and the corresponding fiber orientation tensor, $\mathbf{A}$.

2.1 Tensor based orientation models

Over the last four decades, several tensor-based rotary diffusion models have been proposed. In the present work, we focus on models we consider to be more impactful based on recent reviews and textbooks [6, 7, 10]. The interested reader is referred to the literature for a more in-depth review and comparison of different models.

The choice of an anisotropic rotary diffusion (ARD) model depends on the fiber characteristics and the flow kinematics. The most widely used models in the literature [1, 4, 11–14] are all phenomenological in nature, without a definitive consensus on which yields the most accurate predictions. Some guidance on how to choose the ARD model is provided in [6]. Given the

anisotropic character of the rotary diffusion tensor, its directional properties must be inferred from direction-dependent quantities in the system [6]. Typically, $\mathbf{C} = f(\mathbf{A}, \mathbf{D})$, however, the rotary diffusion may also depend on additional state variables and parameters relevant to the suspension.

The simplest ARD model is the isotropic rotary diffusion (IRD) model, Equation (2.8), proposed by Folgar and Tucker [3]. It models fiber-fiber interactions with uniform strength in all directions through a phenomenological constant C_I.

$$\mathbf{C}^{\mathrm{IRD}} = C_I \mathbf{I} \tag{2.8}$$

When an experimentally determined C_I is unavailable, some authors use semi-empirical relations [15, 16] based on the product of the fiber volume fraction and the fiber aspect-ratio.

Assuming an isotropic contribution for $\mathbf{C}$, is only useful for short-fiber thermoplastics (fiber length $\sim 0.2-0.4$ mm). For long-fiber thermoplastics, (fiber length $\sim 10-13$ mm), the anisotropic character must be explicitly modeled [1]. In the original development of ARD framework, Phelps and Tucker [1] proposed the rotary diffusion tensor as a second-order polynomial with 5 scalar constants. The model reads:

$$\mathbf{C}^{\mathrm{poly}} = b_1 \mathbf{I} + b_2 \mathbf{A} + b_3 \mathbf{A}^2 + \frac{b_4}{\dot{\gamma}} \mathbf{D} + \frac{b_5}{\dot{\gamma}^2} \mathbf{D}^2 \tag{2.9}$$

Although this form can fit experimental data well, certain choices of the coefficients b_i can render the evolution equation of $\mathbf{A}$ unstable and lead to non-physical solutions [6].

To address some of the limitations of the $\mathbf{C}^{\mathrm{poly}}$ model, Tseng and co-workers [12] proposed the improved anisotropic rotary diffusion (iARD) model, where $\mathbf{C}$ is no longer depends on the orientation state, and reads:

$$\mathbf{C}^{\mathrm{iARD}} = C_I^{\mathrm{iARD}} \left(\mathbf{I} - C_M \frac{\mathbf{D}^2}{\|\mathbf{D}^2\|} \right), \tag{2.10}$$

where C_I^{iARD} is a parameter equivalent to C_I in the IRD model, and C_M is a scalar phenomenological parameter representing the fiber-matrix interaction and controlling the degree of anisotropy. Although the model contains only two scalar parameters, it remains stable for a wider range of conditions when compared to $\mathbf{C}^{\mathrm{poly}}$ [6]. The parameter C_M typically has a value in $0 \leq C_M \leq 1$, with $C_M = 0$ recovering the IRD model, and larger values increase the degree of anisotropy. The iARD model is known to be well behaved in simple-shear but may yield non-physical results in some extensional flows if $C_M > 3/4$ [6, 7]. Additionally, this model can be recovered from the second order polynomial model by setting, b_2, b_3 and b_4 to zero.

Further alternatives to model $\mathbf{C}$ were made in 2018 by Tseng et al. [13] and Bakharev et al. [14]. These closely related models considered $\mathbf{C}$ to be coaxial with $\mathbf{A}$. Given the spectral decompositions of the second-order orientation:

$$\mathbf{A} = \lambda_1 \left(\mathbf{e}_1 \otimes \mathbf{e}_1\right) + \lambda_2 \left(\mathbf{e}_2 \otimes \mathbf{e}_2\right) + \lambda_3 \left(\mathbf{e}_3 \otimes \mathbf{e}_3\right), \tag{2.11}$$

where λ_i is the i^{th} eigenvector of $\mathbf{A}$ sorted in descending manner and $\mathbf{e}_i$ is the corresponding eigenvector. For convenience of notation, the tensor product symbol '$\otimes$', will be omitted in what follows. Tseng et al. [13] proposed the principal ARD (pARD) , in which the rotary diffusion tensor is defined as [7]:

$$\mathbf{C}^{\text{pARD}} = C_I^{\text{pARD}} \left(\mathbf{e}_1\,\mathbf{e}_1 + \Omega\,\mathbf{e}_2\,\mathbf{e}_2 + (1-\Omega)\,\mathbf{e}_3\,\mathbf{e}_3\right), \tag{2.12}$$

where $0.5 \leq \Omega \leq 1$ is a fitted scalar parameter. Bakharev et al. [14] proposed the Moldflow rotational diffusion model (MRD), where the rotary diffusion tensor is defined as [7]:

$$\mathbf{C}^{\text{MRD}} = C_I^{\text{MRD}} \left(D_1\,\mathbf{e}_1\,\mathbf{e}_1 + D_2\,\mathbf{e}_2\,\mathbf{e}_2 + D_3\,\mathbf{e}_3\,\mathbf{e}_3\right), \tag{2.13}$$

where D_1, D_2 and D_3 are non-negative scalar quantities.

Both $\mathbf{C}^{\text{pARD}}$ and $\mathbf{C}^{\text{MRD}}$ have a similar formulation, however, $\mathbf{C}^{\text{pARD}}$ is less flexible as it cannot be reduced to the IRD model, whereas the $\mathbf{C}^{\text{MRD}}$ model recovers the IRD by setting all parameters equal to unity. The report of Bakharev et al. [14] provides suitable choices for the model parameters to achieve the best correlation with experimental data.

The choice of ARD model is typically calibrated for some steady-state orientation state, however, their usage in IM simulations must also capture transient behavior accurately. Different studies have shown that the orientation kinetics predicted by the IRD model evolve much faster than the ones observed experimentally [1]. To improve agreement with transient experimental observations, Wang et al. [11, 17] proposed a new objective model (independent of the considered coordinate system) designated reduced-strain closure (RSC)). In this approach, the evolution equation of $\mathbf{A}$ is decomposed into evolution equations for the eigenvalues and eigenvectors. The evolution of the eigenvalues is slowed by introducing a phenomenological scalar parameter ($\kappa \leq 1$), while the evolution equation of the eigenvectors are left unchanged [6]. Using the ARD framework, the evolution equation of the second-order tensor, when using the RSC model reads:

$$\dot{\mathbf{A}}^{\mathrm{J}}_{\mathrm{RSC}} = \mathbf{W}\cdot\mathbf{A} - \mathbf{A}\cdot\mathbf{W} + \xi\left(\mathbf{D}\cdot\mathbf{A} + \mathbf{A}\cdot\mathbf{D} - 2\left[\mathbb{A} + \mathbb{A}_{\mathrm{corr}}\right] : \mathbf{D}\right) \tag{2.14a}$$

$$\mathbb{A}_{\mathrm{corr}} = (1-\kappa)\left(\mathbb{L} - \mathbb{M} : \mathbb{A}\right) \tag{2.14b}$$

$$\dot{\mathbf{A}}^{\mathrm{D}}_{\mathrm{RSC}} = \begin{array}{l}\dot{\gamma}\big[2\left(\mathbf{C} - (1-\kappa)\,\mathbb{M} : \mathbf{C}\right) - 2\kappa\,\mathrm{tr}\left(\mathbf{C}\right)\,\mathbf{A} \\ \qquad - 5\left(\mathbf{C}\cdot\mathbf{A} + \mathbf{A}\cdot\mathbf{C}\right) + 10\left(\mathbb{A} + \mathbb{A}_{\mathrm{corr}}\right) : \mathbf{C}\big]\end{array} \tag{2.14c}$$

$$\mathbb{L} = \sum_{i=1}^{3} \lambda_i \left(\mathbf{e}_i\,\mathbf{e}_i\,\mathbf{e}_i\,\mathbf{e}_i\right), \tag{2.14d}$$

$$\mathbb{M} = \sum_{i=1}^{3} \left(\mathbf{e}_i\,\mathbf{e}_i\,\mathbf{e}_i\,\mathbf{e}_i\right). \tag{2.14e}$$

An equivalent approach to slow-down the orientation kinetics was proposed by Tseng et al. [18] termed the retarding principal rate (RPR). Like the RSC, the RPR methodology slows the evolution of the eigenvalues whilst keeping the eigenvectors unchanged. The key difference lies in the how the correction is introduced. Algorithm 1 summarizes the RPR correction. In brief, the procedure starts by computing the time rate of change of $\mathbf{A}$ of a base model, then based on the current value of $\mathbf{A}$, perform the spectral decomposition as in Equation (2.11). The corresponding eigenvectors are assembled as columns of a rotation matrix $\mathbf{R}$, which is used to rotate $\dot{\mathbf{A}}$ into the principal frame of orientation. The slow-down of the eigenvalues is given via a diagonal matrix, $\dot{\mathbf{\Lambda}}$, computed from the diagonal entries in that frame [12, 18]:

$$\dot{\Lambda}_{ii} = \alpha\left[\lambda_i - \beta\left(\dot{\lambda}_i^2 + 2\dot{\lambda}_j\dot{\lambda}_k\right)\right], \quad i,j,k = 1,2,3 \tag{2.15}$$

where α is a strong slow-down parameter and β is a fine tuning parameter. The resulting correction in then added to the governing orientation model.

Algorithm 1 RPR correction using iARD as an example

1: Compute the baseline $\dot{\mathbf{A}}$:
2: $\quad \dot{\mathbf{A}}^{\mathrm{base}} \leftarrow \dot{\mathbf{A}}^{\mathrm{J}} + \dot{\mathbf{A}}^{\mathrm{D}}\left(\mathbf{C}^{\mathrm{iARD}}\right)$
3: Spectral decomposition of $\mathbf{A}$ (eigenvectors ordered by decreasing eigenvalue):
4: $\quad \mathbf{A} = \mathbf{R}\,\mathbf{\Lambda}\,\mathbf{R}^{\mathrm{T}} \quad$ with $\lambda_1 \geq \lambda_2 \geq \lambda_3$.
5: Rotate the baseline time rate of change to the principal frame of $\mathbf{A}$:
6: $\quad \bar{\dot{\mathbf{\Lambda}}} \leftarrow \mathbf{R}^{\mathrm{T}}\,\dot{\mathbf{A}}^{\mathrm{base}}\,\mathbf{R}$
7: Take the principal (diagonal) values of $\bar{\dot{\mathbf{\Lambda}}}$, and apply a correction (slow-down factor).
8: $\quad$ Equation (2.15) or equivalent
9: Rotate $\bar{\dot{\mathbf{\Lambda}}}$ to laboratory frame to apply as a correction to the baseline $\dot{\mathbf{A}}$:
10: $\quad \dot{\mathbf{A}}^{\mathrm{RPR}} \leftarrow -\mathbf{R}\cdot\bar{\dot{\mathbf{\Lambda}}}\cdot\mathbf{R}^{\mathrm{T}}$
11: Compute the final orientation rate:
12: $\quad \dot{\mathbf{A}} \leftarrow \dot{\mathbf{A}}^{\mathrm{base}} + \dot{\mathbf{A}}^{\mathrm{RPR}}$

Other fiber orientation models were proposed in the literature, such as Wang's two constant model [6], Kugler et al. [19] flow dependent model, MRD/pARD with RSC formulation for slowing fiber orientation kinetics from Kugler et al. [20] and Kech et al. [21], and even data-driven approaches [22, 23]. The interested reader is referred to [6, 7, 10] for a comprehensive literature review of the most relevant associated topics.

The reader is also advised that some models are protected by patents, such as the case for the iARD, pARD and RPR of CoreTech System Co., Ltd [24–28]. Although these approaches are widely discussed in the scientific literature and textbooks, implementation details and scrutiny may be constrained by the enforcement of intellectual property rights. As a result, these models pose constraints on academic dissemination due to the risk of patent infringement.

2.2 Closures

One of the drawbacks of using the tensor-based formulation, as a macro-descriptor of fiber orientation, is the need to calculate the next even-ordered orientation tensor, which results from the derivation of the evolution equation for the lower order tensor. Breuer et al. [29] identified in the literature 17 closures to describe A_{ijkl} and 4 to identify a sixth-order tensor. However, in practice, only the hybrid, the invariant-based optimal fitting (IBOF), or an orthotropic fitted closure are typically used [6, 10].

The hybrid closure was described in the review of Advani and Tucker [30] as being a blend between a quadratic and a linear closure. For a 3D case, the model reads:

$$A_{ijkl}^{\mathrm{hyb}} = f A_{ijkl}^{\mathrm{quad}} + (1 - f)\, A_{ijkl}^{\mathrm{lin}}, \tag{2.16}$$

where f is the blending factor, typically computed as $f = 1 - 27 \det\left(A_{ij}\right)$ [30]. A_{ijkl}^{quad} is computed with Equation (2.17), and A_{ijkl}^{lin} with Equation (2.18).

$$A_{ijkl}^{\mathrm{quad}} = A_{ij} A_{kl}, \tag{2.17}$$

$$\begin{aligned} A_{ijkl}^{\mathrm{lin}} = &- \frac{1}{35}\left(\delta_{ij}\delta_{kl} + \delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right) \\ &+ \frac{1}{7}\left(A_{ij}\delta_{kl} + A_{ik}\delta_{jl} + A_{il}\delta_{jk} + A_{jl}\delta_{ik} + A_{jk}\delta_{il} + A_{kl}\delta_{ij}\right) \end{aligned} \tag{2.18}$$

with δ_{ij} being the second-order identity tensor.

The quadratic closure component is known to be exact when the fibers are perfectly aligned in a single direction, while the linear closure component is known to be exact for isotropic orientation [30].

This blended closure is used due to its low computational cost, when compared to other alternatives. However, it is known to over-predict flow-induced fiber orientation [31, 32].

The eigenvalue-based version of the natural closure (ORE) [33] is an orthotropic eigenvalue-based fitted closure. Due to the normalization condition, Equation. (2.2), the sum of the eigenvalues of $\mathbf{A}$ must equal unity. Consequently, only two eigenvalues are independent, typically λ_1 and λ_2.

Consider $\bar{\mathbb{A}}$ to be the fourth-order tensor $\mathbb{A}$ expressed in the principal axis of $\mathbf{A}$. Due to the assumed orthotropic character, this tensor can be written in contracted notation having 12 independent parameters, as [6, 10, 31]:

$$\bar{\mathbb{A}} = \begin{bmatrix} \bar{\mathbb{A}}_{11} & \bar{\mathbb{A}}_{12} & \bar{\mathbb{A}}_{13} & 0 & 0 & 0 \\ \bar{\mathbb{A}}_{21} & \bar{\mathbb{A}}_{22} & \bar{\mathbb{A}}_{23} & 0 & 0 & 0 \\ \bar{\mathbb{A}}_{31} & \bar{\mathbb{A}}_{32} & \bar{\mathbb{A}}_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & \bar{\mathbb{A}}_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & \bar{\mathbb{A}}_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & \bar{\mathbb{A}}_{66} \end{bmatrix}. \tag{2.19}$$

Assuming a complete symmetry of the fourth-order tensor, there are 6 independent variables, given that:

$$\begin{aligned} \bar{\mathbb{A}}_{23} &= \bar{\mathbb{A}}_{32} = \bar{\mathbb{A}}_{44}, \\ \bar{\mathbb{A}}_{13} &= \bar{\mathbb{A}}_{31} = \bar{\mathbb{A}}_{55}, \\ \bar{\mathbb{A}}_{12} &= \bar{\mathbb{A}}_{21} = \bar{\mathbb{A}}_{66}. \end{aligned} \tag{2.20}$$

Moreover, taking into account that the higher-order tensor, A_{ijkl}, includes information from the lower-order tensor, since $A_{ijkk} = A_{ij}$ [6, 10], the number of independent components can be further reduced to 3, by:

$$\begin{aligned} \bar{\mathbb{A}}_{11} + \bar{\mathbb{A}}_{12} + \bar{\mathbb{A}}_{13} &= \lambda_1, \\ \bar{\mathbb{A}}_{21} + \bar{\mathbb{A}}_{22} + \bar{\mathbb{A}}_{23} &= \lambda_2, \\ \bar{\mathbb{A}}_{31} + \bar{\mathbb{A}}_{32} + \bar{\mathbb{A}}_{33} &= \lambda_3, \end{aligned} \tag{2.21}$$

and

$$\begin{aligned} \bar{\mathbb{A}}_{11} &= \mathrm{f}_1(\lambda_1, \lambda_2), \\ \bar{\mathbb{A}}_{22} &= \mathrm{f}_2(\lambda_1, \lambda_2), \\ \bar{\mathbb{A}}_{33} &= \mathrm{f}_3(\lambda_1, \lambda_2), \end{aligned} \tag{2.22}$$

where f_1, f_2, and f_3 are functions of the independent eigenvalues of $\mathbf{A}$. For the ORE closure, these are defined as:

$$\begin{aligned}\mathrm{f}_i(\lambda_1,\lambda_2) = C_{i_1} &\\ &+ \left(C_{i_2}\lambda_1 + C_{i_3}\lambda_2\right)\\ &+ \left(C_{i_4}\lambda_1\lambda_2 + C_{i_5}\lambda_1^2 + C_{i_6}\lambda_2^2\right) \qquad i \in \{1,2,3\}.\\ &+ \left(C_{i_7}\lambda_1^2\lambda_2 + C_{i_8}\lambda_1\lambda_2^2 + C_{i_9}\lambda_1^3 + C_{i_{10}}\lambda_2^3\right)\\ &+ \left(C_{i_{11}}\lambda_1^2\lambda_2^2 + C_{i_{12}}\lambda_1^3\lambda_2 + C_{i_{13}}\lambda_1\lambda_2^3 + C_{i_{14}}\lambda_1^4 + C_{i_{15}}\lambda_2^4\right),\end{aligned} \tag{2.23}$$

The coefficients $C_{i1}, \ldots, C_{i15}$ are reported in Verweyst's PhD thesis [33] but are also available in [34].

Considering the normalization condition, the entries for $\bar{\mathbb{A}}_{44}$, $\bar{\mathbb{A}}_{55}$ and $\bar{\mathbb{A}}_{66}$ can be computed as [33]:

$$\begin{aligned}\bar{\mathbb{A}}_{44} &= \frac{1}{2}\left(1 - 2\lambda_1 + \bar{\mathbb{A}}_{11} - \bar{\mathbb{A}}_{22} - \bar{\mathbb{A}}_{33}\right),\\ \bar{\mathbb{A}}_{55} &= \frac{1}{2}\left(1 - 2\lambda_2 - \bar{\mathbb{A}}_{11} + \bar{\mathbb{A}}_{22} - \bar{\mathbb{A}}_{33}\right),\\ \bar{\mathbb{A}}_{66} &= \frac{1}{2}\left(-1 + 2\lambda_1 + 2\lambda_2 - \bar{\mathbb{A}}_{11} - \bar{\mathbb{A}}_{22} + \bar{\mathbb{A}}_{33}\right).\end{aligned} \tag{2.24}$$

Once the components of $\bar{\mathbb{A}}$ have been calculated, the tensor is rotated to the laboratory frame by:

$$A_{ijkl} = R_{im}R_{jn}R_{ko}R_{lp}\bar{\mathbb{A}}_{mnop}. \tag{2.25}$$

Another commonly used alternative is the Invariant-Based Optimal Fitting closure, proposed by Chung and Kwon [32]. This closure does not require the spectral decomposition of $\mathbf{A}$ and, instead, relies on the invariants of the second-order orientation tensor. The IBOF closure reads [32]:

$$\begin{aligned}A_{ijkl} = \beta_1\,\mathrm{sym}\left(\delta_{ij}\delta_{kl}\right) + \beta_2\,\mathrm{sym}\left(\delta_{ij}A_{kl}\right)&\\ + \beta_3\,\mathrm{sym}\left(A_{ij}A_{kl}\right) + \beta_4\,\mathrm{sym}\left(\delta_{ij}A_{km}A_{ml}\right)&\\ + \beta_5\,\mathrm{sym}\left(A_{ij}A_{km}A_{ml}\right) + \beta_6\,\mathrm{sym}\left(A_{im}A_{mj}A_{kn}A_{nl}\right),&\end{aligned} \tag{2.26}$$

where the function $\mathrm{sym}\,(\cdot)$ gives the fully symmetric part of its fourth-order tensor argument:

$$\begin{aligned}\mathrm{sym}(T_{ijkl}) = \frac{1}{24}\Big[&T_{ijkl} + T_{ijlk} + T_{ikjl} + T_{iklj} + T_{iljk} + T_{ilkj}\\ &+ T_{jikl} + T_{jilk} + T_{jkil} + T_{jkli} + T_{jlik} + T_{jlki}\\ &+ T_{kijl} + T_{kilj} + T_{kjil} + T_{kjli} + T_{klij} + T_{klji}\\ &+ T_{lijk} + T_{likj} + T_{ljik} + T_{ljki} + T_{lkij} + T_{lkji}\Big].\end{aligned} \tag{2.27}$$

The six β_i coefficients are functions of the second (I_2) and third (I_3) invariants of $\mathbf{A}$, given by:

$$I_2 = \frac{1}{2}\left[\mathrm{tr}\left(\mathbf{A}\right)^2 - \mathrm{tr}\left(\mathbf{A}^2\right)\right], \tag{2.28}$$

$$I_3 = \det\left(\mathbf{A}\right). \tag{2.29}$$

Additionally, only three coefficients are independent, β_3, β_4 and β_6, whose values are given by:

$$\begin{aligned}\beta_i = \; & a_{i1} \\ & + \left(a_{i2}I_2 + a_{i4}I_3\right) \\ & + \left(a_{i3}I_2^2 + a_{i6}I_2I_3 + a_{i5}I_3^2\right) \\ & + \left(a_{i9}I_2^3 + a_{i7}I_2^2I_3 + a_{i8}I_2I_3^2 + a_{i10}I_3^3\right) \\ & + \left(a_{i14}I_2^4 + a_{i11}I_2^3I_3 + a_{i12}I_2^2I_3^2 + a_{i13}I_2I_3^3 + a_{i15}I_3^4\right) \\ & + \left(a_{i20}I_2^5 + a_{i16}I_2^4I_3 + a_{i17}I_2^3I_3^2 + a_{i18}I_2^2I_3^3 + a_{i19}I_2I_3^4 + a_{i21}I_3^5\right),\end{aligned} \qquad i \in \{3,4,6\}. \tag{2.30}$$

where the fitting coefficients $a_{i1},\ldots,a_{i21}$ are provided in the Appendix of [32]. The remaining coefficients are formulated as [32]:

$$\begin{aligned}\beta_1 = & \frac{3}{5}\left[-\frac{1}{7} + \frac{1}{5}\beta_3\left(\frac{1}{7} + \frac{4}{7}I_2 + \frac{8}{3}I_3\right) - \beta_4\left(\frac{1}{5} - \frac{8}{15}I_2 - \frac{14}{15}I_3\right)\right. \\ & \left. -\beta_6\left(\frac{1}{35} - \frac{24}{105}I_3 - \frac{4}{35}I_2 + \frac{16}{15}I_2I_3 + \frac{8}{35}I_2^2\right)\right],\end{aligned} \tag{2.31}$$

$$\beta_2 = \frac{6}{7}\left[1 - \frac{1}{5}\beta_3\left(1 + 4I_2\right) + \frac{7}{5}\beta_4\left(\frac{1}{6} - I_2\right) - \beta_6\left(-\frac{1}{5} + \frac{2}{3}I_3 + \frac{4}{5}I_2 - \frac{8}{5}I_2^2\right)\right], \tag{2.32}$$

$$\beta_5 = -\frac{4}{5}\beta_3 - \frac{7}{5}\beta_4 - \frac{6}{5}\beta_6\left(1 - \frac{4}{3}I_2\right). \tag{2.33}$$

Due to the reduced number of operations compared to eigen-fitted closure, the IBOF closure is considered computationally more efficient, while maintaining a similar level of accuracy [32].

References

[1] Jay H. Phelps and Charles L. Tucker. An anisotropic rotary diffusion model for fiber orientation in short- and long-fiber thermoplastics. *Journal of Non-Newtonian Fluid Mechanics*, 156(3):165–176,

2009.

[2] Jeffery George Barker. The motion of ellipsoidal particles immersed in a viscous fluid. *Proc. R. Soc. Lond. A*, 102:161–179, 1922.

[3] Fransisco Folgar and Charles L. Tucker. Orientation Behavior of Fibers in Concentrated Suspensions. *Journal of Reinforced Plastics and Composites*, 3(2):98–119, 1984.

[4] Suresh G. Advani and Charles L. Tucker. The use of tensors to describe and predict fiber orientation in short fiber composites. *Journal of Rheology*, 31(8):751–784, 1987.

[5] Julian Karl Bauer and Thomas Böhlke. Variety of fiber orientation tensors. *Mathematics and Mechanics of Solids*, 27(7):1185–1211, 2022.

[6] C.L. Tucker. *Fundamentals of Fiber Orientation: Description, Measurement and Prediction*. Hanser Publishers, Munich, Germany, 2022.

[7] Anthony J. Favaloro and Charles L. Tucker. Analysis of anisotropic rotary diffusion models for fiber orientation. *Composites Part A: Applied Science and Manufacturing*, 126:105605, 2019.

[8] R. Mezher, M. Perez, A. Scheuer, E. Abisset-Chavanne, F. Chinesta, and R. Keunings. Analysis of the Folgar and Tucker model for concentrated fibre suspensions in unconfined and confined shear flows via direct numerical simulation. *Composites Part A: Applied Science and Manufacturing*, 91:388–397, 2016.

[9] Kevin Breuer, Markus Stommel, and Wolfgang Korte. Analysis and evaluation of fiber orientation reconstruction methods. *Journal of Composites Science*, 3(3), 2019.

[10] Susanne Katrin Kugler, Armin Kech, Camilo Cruz, and Tim Osswald. Fiber Orientation Predictions—A Review of Existing Models. *Journal of Composites Science*, 4(2):69, 2020.

[11] Jin Wang, John F. O'Gara, and Charles L. Tucker. An objective model for slow orientation kinetics in concentrated fiber suspensions: Theory and rheological evidence. *Journal of Rheology*, 52(5):1179–1200, 2008.

[12] Huan-Chang Tseng, Rong-Y. Chang, and Chia-Hsiang Hsu. An objective tensor to predict anisotropic fiber orientation in concentrated suspensions. *Journal of Rheology*, 60(2):215–224, 2016.

[13] Huan-Chang Tseng, Rong-Yeu Chang, and Chia-Hsiang Hsu. The use of principal spatial tensor to predict anisotropic fiber orientation in concentrated fiber suspensions. *Journal of Rheology*, 62(1):313–320, 2018.

[14] Bakharev Alexander, Yu Huagang, Ray Shishir, Speight Russell, and Jin Wang. Using new anisotropic rotational diffusion model to improve prediction of short fibers in thermoplastic injection molding. In *SPE ANTEC Conference*, 2018.

[15] A. J. Pontes, N. M. Neves, and A. S. Pouzada. The role of the interaction coefficient in the prediction of the fiber orientation in planar injection moldings. *Polymer Composites*, 24(3):358–366, 2003.

[16] Florian Wittemann, Robert Maertens, Luise Kärger, and Frank Henning. Injection molding simulation of short fiber reinforced thermosets with anisotropic and non-Newtonian flow behavior. *Composites Part A: Applied Science and Manufacturing*, 124:105476, 2019.

[17] Jin Wang. *Improved Fiber Orientation Predictions for Injection Molded Composites*. PhD thesis, University of Illinois at Urbana-Champaign, 2007.

[18] Huan-Chang Tseng, Rong-Yeu Chang, and Chia-Hsiang Hsu. Phenomenological improvements to predictive models of fiber orientation in concentrated suspensions. *Journal of Rheology*, 57(6):1597–1631, 2013.

[19] Susanne Katrin Kugler, Argha Protim Dey, Sandra Saad, Camilo Cruz, Armin Kech, and Tim Osswald. A flow-dependent fiber orientation model. *Journal of Composites Science*, 4(3), 2020.

[20] Susanne K. Kugler, Gregory M. Lambert, Camilo Cruz, Armin Kech, Tim A. Osswald, and Donald G. Baird. Efficient parameter identification for macroscopic fiber orientation models with experimental data and a mechanistic fiber simulation. *AIP Conference Proceedings*, 2205:1–6, 2020.

[21] Armin Kech, Susanne Kugler, and Tim Osswald. Significance of model parameter variations in the PARD-RSC model. *Journal of Composites Science*, 4(3), 2020.

[22] Adrien Scheuer, Amine Ammar, Emmanuelle Abisset-Chavanne, Elias Elías Elias Cueto, Adrien Scheuer, Amine Ammar, Emmanuelle Abisset-Chavanne, Elias Elías Elias Cueto, Francisco Chinesta, Roland Keunings, and Suresh G. Advani. Data-driven upscaling of orientation kinematics in suspensions of rigid fibres. *Computer Modeling in Engineering and Sciences*, 117(3):367–386, 2018.

[23] Minyoung Yun, Clara Argerich Martin, Pierre Giormini, Francisco Chinesta, and Suresh Advani. Learning the macroscopic flow model of short fiber suspensions from fine-scale simulated data. *Entropy*, 22(1):30, 2020.

[24] Huan Chang Tseng, Rong Yeu Chang, and Chia Hsiang Hsu. Method and computer readable media for determining orientation of fibers in a fluid. US Patent US8571828B2, 2013.

[25] Huan Chang Tseng, Chia Hsiang Hsu, and Rong Yeu Chang. Computer-implemented simulation method and non-transitory computer medium capable of predicting fiber orientation for use in a molding process. US Patent US9283695B1, 2016.

[26] Huan Chang Tseng, Rong Yeu Chang, and Chia Hsiang Hsu. Molding system for preparing an injection molded fiber reinforced composite article. US Patent US9919465B1, 2018.

[27] Huan Chang Tseng, Rong Yeu Chang, and Chia Hsiang Hsu. Method and computer readable media for determining orientation of fibers in a fluid. EP Patent EP2537661B1, 2016.

[28] Huan Chang Tseng, Rong Yeu Chang, and Chia Hsiang Hsu. Method and computer readable media for determining orientation of fibers in a fluid comprising a polymer. EP Patent EP2679377B1, 2016.

[29] Kevin Breuer, Markus Stommel, and Wolfgang Korte. Analysis and evaluation of fiber orientation reconstruction methods. *Journal of Composites Science*, 3(3), 2019.

[30] Suresh G. Advani and Charles L. Tucker. Closure approximations for three‐dimensional structure tensors. *Journal of Rheology*, 34(3):367–386, 1990.

[31] Jr. Cintra, Joaquim S. and III Tucker, Charles L. Orthotropic closure approximations for flow‐induced fiber orientation. *Journal of Rheology*, 39(6):1095–1122, 11 1995.

[32] Du Hwan Chung and Tai Hun Kwon. Invariant-based optimal fitting closure approximation for the numerical prediction of flow-induced fiber orientation. *Journal of Rheology*, 46(1):169–194, 2002.

[33] Brent Eugene VerWeyst. *Numerical Predictions of Flow-Induced Fiber Orientation in Three-Dimensional Geometries*. PhD thesis, University of Illinois at Urbana-Champaign, 1998.

[34] Fiber orientation tools. `https://github.com/charlestucker3/Fiber-Orientation-Tools`. Accessed: 2024-02-27.

Chapter 3 Physics-Informed data-driven closure relation for dilute short fiber suspensions

Abstract:

Tensor based formulations for modeling fiber orientation require the use of a closure approximation to solve the macro-descriptor evolution equation. This originates from the the hydrodynamic component of motion and has received some attention in literature.

In this work, we use the concept of universal ordinary differential equations to infer a data-driven closure for fibers in the diluted regime using simple homogeneous flows. The closure can be understood as a orthogonal correction to the linear closure in eigenspace. To simplify and accelerate the training process, empirically determined bounds are used to restrain the machine learning output to physically admissible values.

Three sets of tests are used to assess the performance of the model with simple homogeneous flows not included in the training pool, sequential combination of simple flows, and a non-homogeneous flow in a center gated disk.

The results show that the trained model is able to adequately replicate the true dynamics of orientation, even for unseen flow regimes.

Keywords: Machine Learning; Physics informed; Fiber orientation modeling

This chapter was adapted from Ramoa, B., Ghnatios, C., Nóbrega, J. M., & Chinesta, F. (2025). Physics-informed data-driven closure relation for dilute short fiber suspensions. Advanced Modeling and Simulation in Engineering Sciences, 12(1). https://doi.org/10.1186/s40323-025-00290-w

3.1 Introduction

Fibers play a key role in the performance of polymer composites, with the final performance partially dependent on the shape, concentration, and the spatial configuration of the reinforcing phase. Due to flow kinematics, these fillers change their initial configuration, resulting in spatial inhomogeneity and anisotropic mechanical properties [1]. Accurate prediction of the fibers orientation state should enhance the precision of mechanical models used to evaluate parts performance [1–3].

Orientation of rigid, cylindrical fibers can be made at different length scales through several orientation descriptors [4]. These vary in the richness of the information and in the computational power required to obtain it.

In engineering practice it is common to use so-called orientation tensors [5]. These carry condensed information from the moments of a probability distribution function describing the orientation state of a population of fibers. However, the usage of these macro-scale descriptors requires a closure relationship to able to solve its evolution equation. This originates from the hydrodynamic component of orientation where the next even order tensor appears in the formulation of the lower order descriptor [5].

In literature, several closure types have been reported [6–12] based on mechanistic models, fitted models and even models using artificial intelligence techniques [10, 11] for concentrated suspensions.

Given the unknown mathematical expression for the closure relation, we propose a systematic approach with the objective of dissecting the effect that originates from the hydrodynamic and diffusion contributions, focusing this work solely on developing the former. To this end, a new data-driven modeling paradigm is available through universal ordinary differential equations (UODEs) , linking universal approximators with differential equations and corresponding solvers. These are a generalization of Neural Ordinary Differential Equations [13], which can contain prior system knowledge and have been successfully used in several areas [14–18]. This approach should provide a closure relation that takes into account the system dynamics, making it a more robust method than the traditional polynomial/multi-layer black-box approaches described in the literature. To the best of the author's knowledge, this methodology is not covered in the available literature regarding orientation modeling.

In this work, we develop a neural network closure based on a UODE for the hydrodynamic component of orientation. Data was generated in the dilute regime to disregard any possible diffusion contributions. The devised closure can be understood as an orthogonal correction to the linear closure in eigenspace. Due to the complexity of the data to be fitted, the initial attempts were unsuccessful. To obtain a more efficient fitting method, the system behavior is studied in detail to identify the physical bounds of the unknown correction term. This knowledge, is, then, used in the innovative approach proposed in this work to restrict the search space, which allowed to efficiently devise a proper data-driven

model.

The remaining paper is organized as follows: Section 3.2 highlights previous works with machine learning technology in flow-induced fiber orientation modeling. Section 3.3, the approach to generate synthetic data for training and testing the UODE. Section 3.4, describes the universal ordinary differential equation methodology and determination of admissible bounds for the closure. Section 3.5 showcases the test cases and results, and finally, Section 3.6 summarizes the main conclusions drawn from this work.

3.2 Previous Data Driven Approaches

The usage of artificial intelligence techniques has been explored in fiber orientation modeling. To the best of the author's knowledge, only two closure relations based on artificial intelligence techniques have been reported, the NNET [11] and NNORT [10]. The former uses the a fully symmetric 4^{th}-order tensor with 15 independent components. In Mandel contracted notation [19], denoted by a tilde, $\mathbb{A}$ is represented as:

$$\tilde{\mathbb{A}} = \begin{bmatrix} \tilde{\mathbb{A}}_{11} & \tilde{\mathbb{A}}_{12} & \tilde{\mathbb{A}}_{13} & \nu\tilde{\mathbb{A}}_{14} & \nu\tilde{\mathbb{A}}_{15} & \nu\tilde{\mathbb{A}}_{16} \\ \tilde{\mathbb{A}}_{12} & \tilde{\mathbb{A}}_{22} & \tilde{\mathbb{A}}_{23} & \nu\tilde{\mathbb{A}}_{24} & \nu\tilde{\mathbb{A}}_{25} & \nu\tilde{\mathbb{A}}_{26} \\ \tilde{\mathbb{A}}_{13} & \tilde{\mathbb{A}}_{23} & \tilde{\mathbb{A}}_{33} & \nu\tilde{\mathbb{A}}_{34} & \nu\tilde{\mathbb{A}}_{35} & \nu\tilde{\mathbb{A}}_{36} \\ \nu\tilde{\mathbb{A}}_{14} & \nu\tilde{\mathbb{A}}_{24} & \nu\tilde{\mathbb{A}}_{34} & 2\tilde{\mathbb{A}}_{23} & 2\tilde{\mathbb{A}}_{36} & 2\tilde{\mathbb{A}}_{25} \\ \nu\tilde{\mathbb{A}}_{15} & \nu\tilde{\mathbb{A}}_{25} & \nu\tilde{\mathbb{A}}_{35} & 2\tilde{\mathbb{A}}_{36} & 2\tilde{\mathbb{A}}_{13} & 2\tilde{\mathbb{A}}_{14} \\ \nu\tilde{\mathbb{A}}_{16} & \nu\tilde{\mathbb{A}}_{26} & \nu\tilde{\mathbb{A}}_{36} & 2\tilde{\mathbb{A}}_{25} & 2\tilde{\mathbb{A}}_{14} & 2\tilde{\mathbb{A}}_{12} \end{bmatrix}. \tag{3.1}$$

$$\nu = \sqrt{2}$$

For sake of clarity, the relation between the regular and contracted Mandel notation for $\mathbb{A}$ components is given in the Appendix A.

The number of independent parameters can be further reduced by considering the contraction property, $A_{ijkk} = A_{ij}$, which decreases the number of independent components of $\mathbb{A}$ to 9.

The authors of the NNET [11] used a fully connected, feed-forward neural network with 5 inputs (independent components of $\mathbf{A}_2$), 1 hidden layer with 95 neurons and an output layer to predict the 9 independent components of $\mathbb{A}$. A hyperbolic tangent was used as a non-linear activation function for the first layer and the linear/identity function for the output layer. The resulting closure reads:

$$\mathbb{A} = \mathbf{w}_2 \tanh\left(\mathbf{w}_1 \mathbf{A}_{2_{\text{input}}} + \mathbf{b}_1\right) + \mathbf{b}_2, \tag{3.2}$$

with $\mathbf{w}_n$ and $\mathbf{b}_n$ being the weights and bias of the n^{th} layer in the network. The training data was obtained by solving the ODF from isotropic state, using the isotropic rotary diffusion model in 3D, Equation (2.8) [20], and a group of 3 simple homogeneous flows (simple-shear, uniaxial elongation, biaxial elongation) and 8 mixed flows (shear-biaxial, shear-planar, shear-uniaxial and shear-stretch) with $C_I = 10^{-2}$.

At each time step, $\mathbf{A}_2$ and $\mathbb{A}$ were computed following Equation (2.6). In this approach, the network provides a map between the computed $\mathbf{A}_2$ and $\mathbb{A}$. The training algorithm, was provided without too many specifics, involved steepest descent, and quasi-Newton methods. However, the resulting procedure is not frame invariant [11] and does not include a mechanism to prevent non-physical predictions of $\mathbb{A}$.

The NNORT network is based on the orthotropic closure mechanism [7], see Section 2.2. The admissible space for each independent eigenvalue is given by [19]:

$$\begin{aligned} \frac{1}{3} \leq \quad & \lambda_1 \leq 1, \\ \frac{1}{2}\left(1-\lambda_1\right) \leq \quad & \lambda_2 \leq \min\left(\lambda_1, 1-\lambda_1\right). \end{aligned} \tag{3.3}$$

This relation is visually represented in Figure 3.1, and is known in literature as the *UBT* triangle [21]. The vertices *U*, *B* and *T* represent, respectively, uniaxial ($\lambda_1 = 1, \lambda_2 = 0$), biaxial ($\lambda_1 = \lambda_2 = \frac{1}{2}$) and triaxial/random ($\lambda_1 = \lambda_2 = \lambda_3 = \frac{1}{3}$) orientation states. The UBT triangle edges also correspond to particular cases, *BU* ($\lambda_2 = 1 - \lambda_1 \wedge \lambda_3 = 0$) contains all planar orientation states, and *TU* ($\lambda_1 > \lambda_2 \wedge \lambda_2 = \lambda_3$) and *TB* ($\lambda_1 = \lambda_2 \wedge \lambda_2 > \lambda_3$) comprise all orientation states that are transversely isotropic [7, 21].

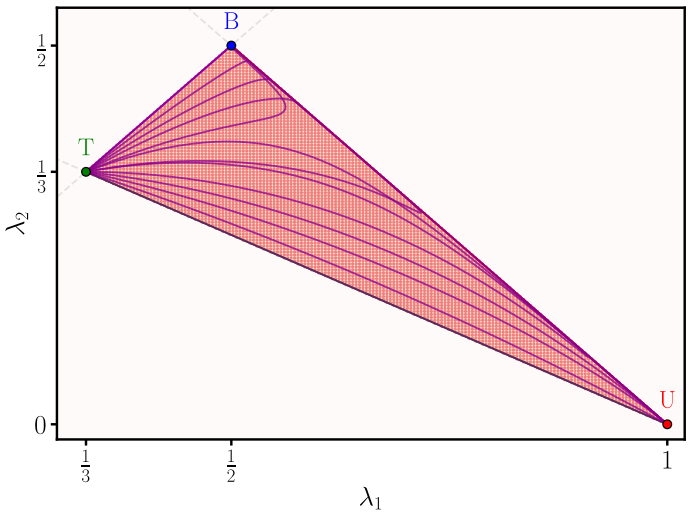


Figure 3.1: Admissible eigenspace of orientation states in terms of eigenvalues.

Considering that the fourth-order tensor is orthotropic when expressed in the principal axes of $\mathbf{A}_2$, i.e., $\bar{\mathbb{A}}^{\text{ortho}}$ contains all directional information from $\mathbf{A}_2$ [7], the number of the associated independent

components can be reduced to 3, based on the properties of the orientation tensors.

The training data used for NNORT was obtained in a similar manner to the NNET, where the evolution equation of the ODF was solved using the IRD model. To cover the admissible eigenspace, 14 representative flows with two levels of C_I were used. The NNORT approach uses a fully connected, feed-forward neural-network with two input variables (independent eigenvalues of $\mathbf{A}_2$), 1 hidden layer with 14 neurons, and 1 output layer giving an estimation for the 3 independent values of $\bar{\mathbb{A}}^{\mathrm{orth}}$. The closure equation has the same format as Equation (3.2). To train the network, the BFGS algorithm was employed, with the mean squared error as a loss function. Since the orthotropic correction is defined in eigenspace, this network is frame invariant.

Although both approaches utilize data with diffusion, they do not prevent the occurrence of nonphysical values of $\mathbb{A}$. This may lead to nonphysical values of $\mathbf{A}_2$ during the integration process, compromising the accuracy and reliability of the obtained results.

During our research, we noticed that training the network without defining admissible bounds to constrain the search space would be exceptionally difficult, and in general would not allow the model to fit and generalize the data coming from a wide range of flow-induced fiber orientation dynamics. In fact, all attempts to train our UODE network prior to the proposed approach proved unsuccessful.

3.3 Data generation

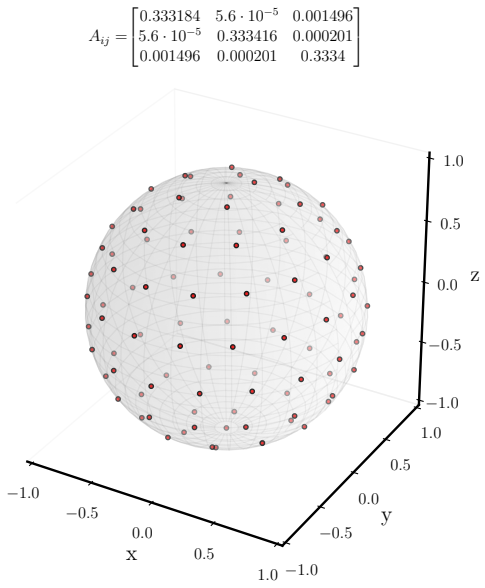
The origin of $\mathbb{A}$ stems from the hydrodynamic contribution, the data to be used in the training procedure could be generated in two ways: On the first approach, one could sample a population of N individual fibers over the unit sphere, e.g., using the algorithm described by Saff and Kuijlaars [22] to generate pairs of (θ, ϕ) angles and solve the hydrodynamic equation of motion, Equation (2.1), for a given incompressible velocity gradient. Afterwards, the second-order tensor can be computed from the discrete samples as:

$$A_{ij} = \frac{1}{N} \sum_{k=1}^{N} p_i^k \, p_j^k. \tag{3.4}$$

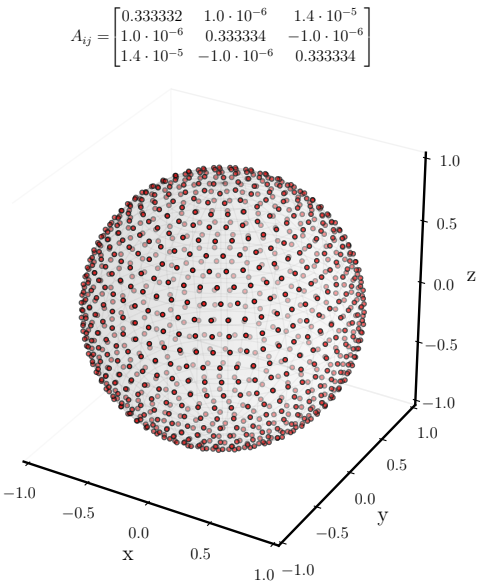
This system constitutes an embarrassingly parallel problem, which can be integrated in time using standard open-source packages [23]. Following the proposed sampling procedure [22], Figure 3.2 shows that with increasing population of fibers, a state close to isotropic can be achieved.

Another alternative to generate data, under the consideration that the initial state is isotropic, is through the solution proposed by Dinh and Armstrong [21, 24], together with the calculation procedure of Smith et al. [21, 25], in which:

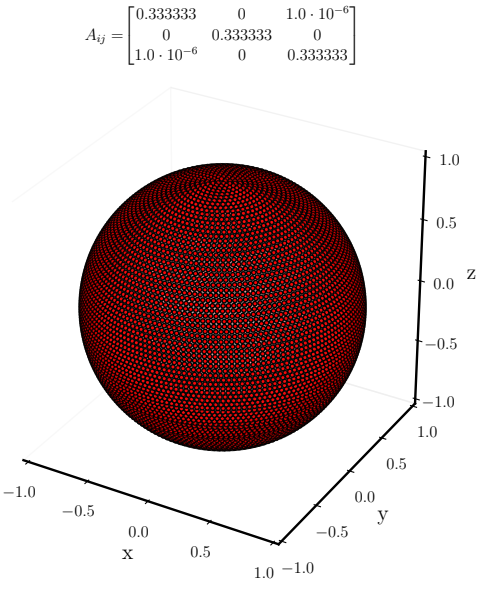
$$\psi\left(\mathbf{p}\right) = \frac{1}{4\pi} \frac{1}{\left(\mathbf{p} \cdot \mathbf{B}^{-1} \cdot \mathbf{p}\right)^{3/2}}, \tag{3.5}$$



(a) $N_{\text{points}} = 100$



(b) $N_{\text{points}} = 1\,000$



(c) $N_{\text{points}} = 10\,000$

Figure 3.2: Sampling of points over the surface of the unit sphere and the corresponding 2^{nd}-order orientation tensor.

where:

- B_{ij}^{-1} is the Piola tensor [21], which is symmetric and has a unitary determinant.

This tensor must satisfy the following differential equation [25]:

$$\begin{aligned} &\frac{\mathrm{D}\mathbf{B}^{-1}}{\mathrm{D}t} = -\mathbf{B}^{-1}\cdot(\mathbf{W}+\xi\mathbf{D}) - (-\mathbf{W}+\xi\mathbf{D})\cdot\mathbf{B}^{-1}, \\ &\text{with } \mathbf{B}^{-1} = \mathbf{I} \text{ at } t = 0 \end{aligned} \tag{3.6}$$

The Piola tensor can be diagonalized as: $\mathbf{B}^{-1} = \mathbf{R}\cdot\mathrm{diag}\,[b_1, b_2, b_3]\cdot\mathbf{R}^{\mathrm{T}}$, with b_i sorted in ascending order. The diagonal elements are then used to compute the eigenvalues of $\mathbf{A}_2$, by:

$$\lambda_1 = \frac{1}{2}\int_0^\infty \frac{\mathrm{d}s}{(b_1+s)^{3/2}\sqrt{b_2+s}\sqrt{b_3+s}}, \tag{3.7a}$$

$$\lambda_2 = \frac{1}{2}\int_0^\infty \frac{\mathrm{d}s}{\sqrt{b_1+s}\,(b_2+s)^{3/2}\sqrt{b_3+s}}, \tag{3.7b}$$

$$\lambda_3 = \frac{1}{2}\int_0^\infty \frac{\mathrm{d}s}{\sqrt{b_1+s}\sqrt{b_2+s}\,(b_3+s)^{3/2}}. \tag{3.7c}$$

Equations (3.7) are one third of Carlon's R_D elliptic integral [25], which can be computed with open-source packages [23, 26]. The second-order orientation tensor can be recovered, using the same similarity transformation, $\mathbf{A}_2 = \mathbf{R}\cdot\mathrm{diag}\,[\lambda_1, \lambda_2, \lambda_3]\cdot\mathbf{R}^{\mathrm{T}}$.

The results plotted in Figure 3.3 allow comparing the temporal evolution of the A_{11} component using both approaches described above for different basic flows. Visually the results are well matched, numerically, there are minor discrepancies that are reduced with increasing population of fibers. The maximum absolute error for the A_{11} component occurs in the uniaxial flow with a value of 4.1×10^{-4}. The computational time is however quite different, as solving Jeffery's equations for a large population takes longer than computing the required diagonalization and solving the elliptical integral. In the experiments performed, generating $1\,000$ data points through population sampling (10^4) in a multi-core computation (8 cores, personal computer) requires approximately 40 s per sample, whilst through the second procedure, tested in single core in the same machine, takes 2 s per sample. Accordingly, in this work, for generating the synthetic data required for training the network, the latter approach was used.

The flows used for training the network are reported in Table 3.1 and where obtained from [10], using a unitary scaling coefficient ($G = 1$). The values of $\mathbf{A}_2$ where computed in the range $t = [0, 10]$ s, with 1000 linearly spaced points and a unitary particle shape factor. The path of each flow on the *UBT* triangle is depicted with purple lines on Figure 3.1.

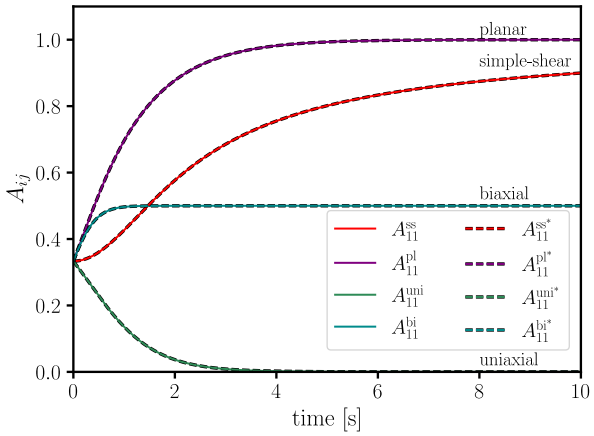


Figure 3.3: Comparison of A_{11} component obtained through [25] (full line), and solving Jeffery's motion for a population of 10^4 fibers (dashed line). Simple-shear (SS), $v = \begin{bmatrix} x_2, & 0, & 0 \end{bmatrix}^{\mathrm{T}}$, planar extension (pl), $v = \begin{bmatrix} x_1, & -x_2, & 0 \end{bmatrix}^{\mathrm{T}}$, uniaxial (uni), $v = \begin{bmatrix} -x_1/2, & -x_2/2, & x_3 \end{bmatrix}^{\mathrm{T}}$, and biaxial (bi), $v = \begin{bmatrix} x_1, & x_2, & -2x_3 \end{bmatrix}^{\mathrm{T}}$

Table 3.1: Flows used for training. G is a scaling parameter.

1. simple shear: $\begin{bmatrix} Gx_3, & 0, & 0 \end{bmatrix}^{\mathrm{T}}$	8. biaxial elongation: $\begin{bmatrix} Gx_1, & Gx_2, & -2Gx_3 \end{bmatrix}^{\mathrm{T}}$
2. uniaxial elongation: $\begin{bmatrix} 2Gx_1, & -Gx_2, & -Gx_3 \end{bmatrix}^{\mathrm{T}}$	9. shear-stretch: $\begin{bmatrix} -Gx_1 + 10Gx_2, & -Gx_2, & 2Gx_3 \end{bmatrix}^{\mathrm{T}}$
3. shear-stretch, $\begin{bmatrix} -Gx_1 + Gx_2, & -Gx_2, & 2Gx_3 \end{bmatrix}^{\mathrm{T}}$	10. shear-stretch: $\begin{bmatrix} -Gx_1 + \frac{15}{4}Gx_2, & -Gx_2, & 2Gx_3 \end{bmatrix}^{\mathrm{T}}$
4. shear-stretch: $\begin{bmatrix} -Gx_1 + \frac{3}{2}Gx_2, & -Gx_2, & 2Gx_3 \end{bmatrix}^{\mathrm{T}}$	11. shear-biaxial: $\begin{bmatrix} Gx_1 + 2Gx_3, & Gx_2, & -2Gx_3 \end{bmatrix}^{\mathrm{T}}$
5. shear-biaxial: $\begin{bmatrix} Gx_1 + \frac{11}{4}Gx_3, & Gx_2, & -2Gx_3 \end{bmatrix}^{\mathrm{T}}$	12. shear-biaxial: $\begin{bmatrix} Gx_1 + \frac{5}{4}Gx_3, & Gx_2, & -2Gx_3 \end{bmatrix}^{\mathrm{T}}$
6. shear-planar: $\begin{bmatrix} -Gx_1 + 10Gx_3, & Gx_2, & 0 \end{bmatrix}^{\mathrm{T}}$	13. shear-planar: $\begin{bmatrix} -Gx_1 + Gx_3, & Gx_2, & 0 \end{bmatrix}^{\mathrm{T}}$
7. shear-uniaxial: $\begin{bmatrix} 2Gx_1 + 3Gx_3, & -Gx_2, & -Gx_3 \end{bmatrix}^{\mathrm{T}}$	

3.4 Universal ordinary differential equations

Universal ordinary differential equations leverage the usage of universal approximators (e.g., neural networks) in ordinary differential equations(ODEs) [13]. If no known dynamics are introduced, the universal approximator can be used as the entire right-hand side of the ODE, resulting in a "black-box" approach, whilst if known dynamics are introduced, the neural-network can capture missing phenomena from the data, which is usually designated as "grey-box" approach [27].

Mathematically, the approach can be represented by:

$$\frac{\mathrm{d}\mathbf{A}_2}{\mathrm{d}t} = S\left(\mathbf{A}_2\right) + NN_\eta\left(\mathbf{A}_2\right), \tag{3.8}$$

where $S\left(\mathbf{A}_2\right)$ represents the known dynamics of the system and NN_η is a neural-network parameterized by η, whose purpose is to capture missing information from the known dynamics [15]. This should reduce the amount of data that is required to train a pure data-driven method [27] and provide better generalization capabilities.

For this work, the UODE was coded in the JAX framework [28] v0.4.23, using the packages: *Diffrax* v0.5.0 [29], *Equinox* v0.11.3 [30], and *Optax* v0.1.9 [31]. Equation (2.7b), was implemented in Mandel contracted notation. At each time step, the 4^{th}-order tensor is computed by a composite layer, Figure 3.4, that takes as input $\mathbf{A}_2$ in Mandel notation, $\tilde{\mathbf{A}}_2 = \left[A_{11},\ A_{22},\ A_{33},\ \sqrt{2}A_{23},\ \sqrt{2}A_{13},\ \sqrt{2}A_{12}\right]^{\mathrm{T}}$. The layer starts by performing the spectral decomposition of $\mathbf{A}$, however, during training it was observed that for some flows, λ_1 could take values higher than one, or that λ_3 could be predicted slightly outside the admissible range. To correct this behavior, λ_1 and λ_2 were projected back into the *UBT* triangle using the "Closest Point on Triangle to Point" algorithm in [32] to force $\mathbf{A}_2$ into an admissible values, λ_1^*, λ_2^*. Afterwards, these are given as input to a linear feed forward network with 1 hidden layer with 10 neurons and an output layer yielding 3 values, ($\hat{d}_1$, $\hat{d}_2$, $\hat{d}_3$), using the hyperbolic tangent as activation function to bound the results, $\hat{d}_i \in [-1, 1]$. Subsequently, the $\hat{d}_i$ values are mapped into admissible ranges of d_1, d_2 and d_3 through a linear map transformation, yielding d_1^*, d_2^*, d_3^*. The network outputs a potentially corrected version of $\mathbf{A}_2$, and $\mathbb{A}$ in Mandel notation, rotated back to laboratory frame using the uncorrected eigenvectors.

3.4.1 Definition of bounds

From the work of Bauer and Böhlke [19], under harmonic decomposition, the fourth-order tensor can be decomposed into isotropic and deviatoric contributions, reading:

$$\mathbb{A} = \mathbb{A}^{\mathrm{ISO}} + \mathbb{A}^{\mathrm{dev2}} + \mathbb{A}^{\mathrm{dev4}} \tag{3.9a}$$

$$\mathbb{A}^{\mathrm{ISO}} = \frac{7}{35}\mathrm{sym}\left(\mathbf{I} \otimes \mathbf{I}\right) \tag{3.9b}$$

$$\mathbb{A}^{\mathrm{dev2}} = \frac{6}{7}\mathrm{sym}\left(\mathrm{dev}\left(\mathbf{A}\right) \otimes \mathbf{I}\right) \tag{3.9c}$$

where:

- $\mathrm{sym}\left(\cdot\right)$ is a function to compute the fully symmetric part of a 4^{th}-order tensor, Equation (2.27) [9].

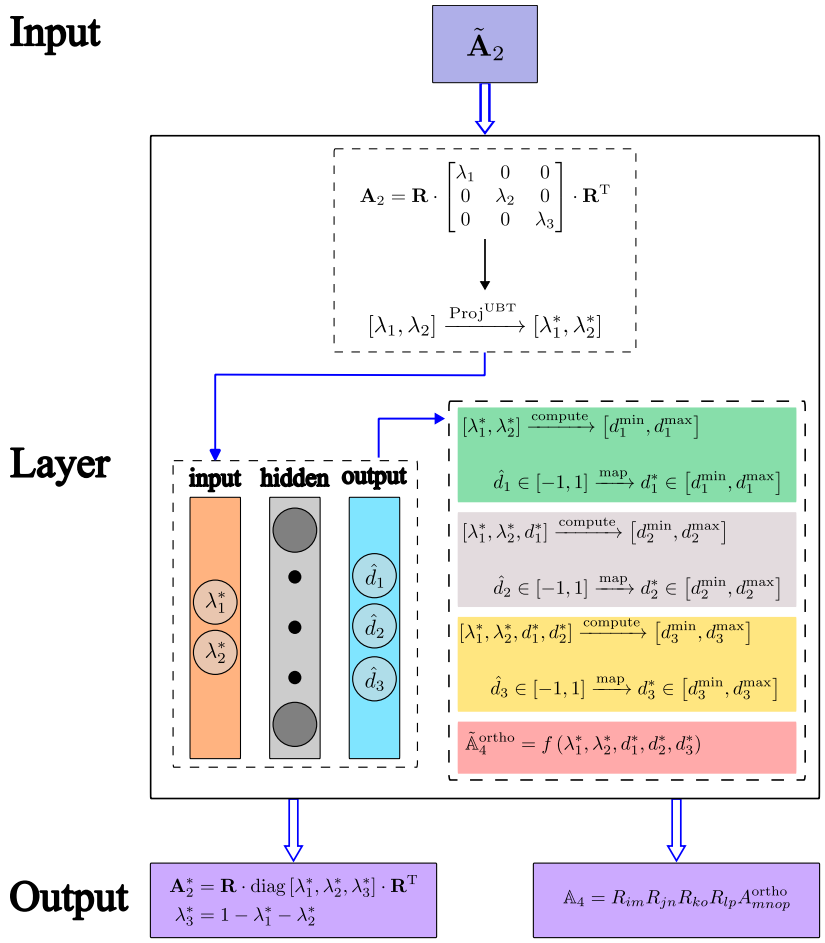


Figure 3.4: Schematic representation of composite layer.

- $\mathrm{dev}\,(\cdot)$ is a function to compute the deviatoric part of a tensor [19].

The term $\mathbb{A}^{\mathrm{dev4}}$, reflects the material symmetry of the ODF and can be parameterized, for the most generic symmetry (triclinic), with $9\ d_i$ scalar parameters [19]. A representation of this fourth-order deviator in contracted Mandel notation is given in Equation (3.10) [19].

$$\bar{\tilde{\mathbb{A}}}^{\mathrm{dev4}} = \begin{bmatrix} -d_1 - d_2 & d_1 & d_2 & -\sqrt{2}\,(d_4 + d_5) & \sqrt{2}d_6 & \sqrt{2}d_8 \\ d_1 & -d_1 - d_3 & d_3 & \sqrt{2}d_4 & -\sqrt{2}\,(d_6 + d_7) & \sqrt{2}d_9 \\ d_2 & d_3 & -d_2 - d_3 & \sqrt{2}d_5 & \sqrt{2}d_7 & -\sqrt{2}\,(d_8 + d_9) \\ -\sqrt{2}\,(d_4 + d_5) & \sqrt{2}d_4 & \sqrt{2}d_5 & 2d_3 & -2d_8 - 2d_9 & -2d_6 - 2d_7 \\ \sqrt{2}d_6 & -\sqrt{2}\,(d_6 + d_7) & \sqrt{2}d_7 & -2d_8 - 2d_9 & 2d_2 & -2d_4 - 2d_5 \\ \sqrt{2}d_8 & \sqrt{2}d_9 & -\sqrt{2}\,(d_8 + d_9) & -2d_6 - 2d_7 & -2d_4 - 2d_5 & 2d_1 \end{bmatrix} \mathbf{B}_{\xi}^{\mathbf{v}} \otimes \mathbf{B}_{\zeta}^{\mathbf{v}}, \quad (3.10)$$

where $\mathbf{B}_{\xi}^{\mathbf{v}}$ and $\mathbf{B}_{\zeta}^{\mathbf{v}}$ are basis vectors spanning the orientation coordinate system in the contracted notation.

This representation of $\mathbb{A}$ highlights isotropy, and the linear closure can be recovered by setting $\mathbb{A}^{\mathrm{dev4}}$ to zero [19]. In the context of this work, Equation (3.9a) can be understood as a correction to the linear closure based on material symmetry in eigenspace. For the orthotropic case, only d_1, d_2,

and d_3 are non-zero. The resulting 6 entries, read:

$$\bar{\bar{\mathbb{A}}}^{\mathrm{ortho}}(\lambda_1,\lambda_2,d_1,d_2,d_3)=\begin{bmatrix}\tilde{\mathbb{A}}_{11}= & -d_1-d_2+\frac{6\lambda_1}{7}-\frac{3}{35}\\ \tilde{\mathbb{A}}_{22}= & -d_1-d_3+\frac{6\lambda_2}{7}-\frac{3}{35}\\ \tilde{\mathbb{A}}_{33}= & -d_2-d_3-\frac{6\lambda_1}{7}-\frac{6\lambda_2}{7}+\frac{27}{35}\\ \tilde{\mathbb{A}}_{23}= & d_3-\frac{\lambda_1}{7}+\frac{4}{35}\\ \tilde{\mathbb{A}}_{13}= & d_2-\frac{\lambda_2}{7}+\frac{4}{35}\\ \tilde{\mathbb{A}}_{12}= & d_1+\frac{\lambda_1}{7}+\frac{\lambda_2}{7}-\frac{1}{35}\end{bmatrix}, \tag{3.11}$$

An equation for admissible parameters of d_1, d_2 and d_3 for the isotropic case $\left(\lambda_1=\lambda_2=\lambda_3=\frac{1}{3}\right)$ is available in Bauer et al. [19] (See Appendix), with the region of admissibility depicted in Figure 3.5.

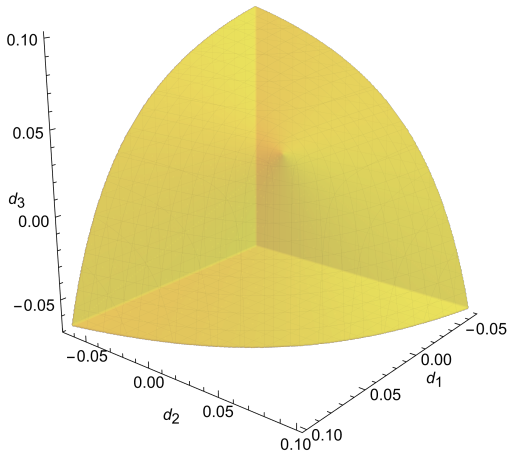


Figure 3.5: Admissible bounds of d_1, d_2 and d_3 for the isotropic state.

To infer possible bounds of d_1, d_2 and d_3, ca. 5500 points across the *UBT* triangle were sampled, red dots on Figure 3.1, and the numerical bounds computed by applying Sylverster's criterion (SC) [33] for positive-definite to $\bar{\bar{\mathbb{A}}}^{\mathrm{ortho}}$ with the entries defined by Equation (3.11) using the software *Mathematica* 14® [34].

By the analysis of the resulting numerical equations it was possible to identify a pattern for the admissibility criteria, with the exception of the points on the *BU* line. The lower bounds, assumed to be included in the admissibility region, are given by the three directly obtainable eigenvalues of $\bar{\bar{\mathbb{A}}}_4$ [19]. Through analysis of the numerical results, the upper bounds, are found from the determinant equation of the third Minor of SC, by solving the equation with respect to d_3. The solution for d_2 is obtained by subtracting the lower to the upper bound of d_3 and solving with respect to d_2. The upper bound for d_1 is computed by taking the previous bound before solving, and replacing the value of d_2 with its lower bound. Afterwards, solving with respect to d_1. The bound equations are given in Appendix B. Concurrently to the development of this work, a similar parametrization was obtained by Kauß et. al. [35].

For the *BU* line, $\mathbf{A}_2$ can be parameterized as a function of only λ_1 since in this region $\lambda_2=$

$1 - \lambda_1$ and $\lambda_3 = 0$. In this line, the orthotropic tensor is degenerated, and the only degree of freedom is related to d_1 [19]. Thus, d_2 and d_3 were fixed at their lower bounds. Using *Mathematica* 14® [34], with SC for positive semi-definite, the admissibility range for $\frac{1}{2} \leq \lambda_1 < 1$, is given by $-\frac{4}{35} \leq d_1 < \frac{1}{35}\left(-4 + 35\lambda_1 - 35\lambda_1^2\right)$ and for $\lambda_1 = 1$, $d_1 = -\frac{4}{35}$.

The bounds determined for d_1, d_2 and d_3 are going to be used to limit the search space of the problem during the learning process of the neural network.

3.4.2 Training

Data obtained as described in Section 3.3 were used in 4 stages to train the model. At each stage the temporal data was increased by 2.5 s while keeping the number of points in the time-series at 250. The number of epochs per stage was [30, 30, 20, 10], respectively.

The UODE was solved using Heun's method (2^{nd}-order explicit Runge–Kutta method) with an absolute and relative tolerance of 10^{-5}. At each training stage the ADAM optimizer was used with a learning-rate of $\left[2 \times 10^{-3},\ 1 \times 10^{-3},\ 5 \times 10^{-4},\ 1 \times 10^{-4}\right]$, respectively. The loss function was calculated after solving Equation (3.8) as the sum of the squared differences between the true (A_{ij}^{true}) and the predicted dynamics (A_{ij}^{pred}) throughout the whole time-series, as follows:

$$\mathcal{L}\left(\eta\right) = \sum_{t} \left(A_{ij}^{\text{true}} - A_{ij}^{\text{pred}}\right)^2. \tag{3.12}$$

The evolution of the average loss throughout the iterative process is depicted in Figure 3.6. For the selected training strategy and data, the average loss function decreases continuously throughout all stages, even when new data is added to the system. At the last training stage, where the whole time series is available, the average loss throughout all flow conditions is ≈ 0.001.

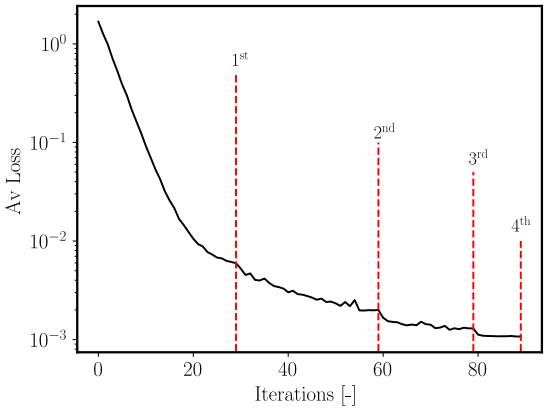


Figure 3.6: Average loss throughout each training epoch.

3.5 Cases and results

In this section the predictive capabilities of the UODE are inferred with three tests cases. These follow the work of Cintra and Tucker [7], which have been also used in other studies [9–11]. The first test uses homogeneous flows that are not included in the training pool. The second test uses a sequence of homogeneous flows. Finally, the last test uses a non-homogeneous flow on a center gated disk (CGD). In all tests the UODE was solved using the same settings as those reported during the training stage. To quantify the accuracy during the transient flows a scalar magnitude of the time varying error [7, 9], $e(t)$, between the true and predicted values of $\mathbf{A}_2$, was calculated as:

$$\mathbf{e}(t) = \mathbf{A}_2^{\text{true}}(t) - \mathbf{A}_2^{\text{pred}}(t), \tag{3.13}$$

$$e(t) = \sqrt{\frac{1}{2} e_{ij} e_{ji}}. \tag{3.14}$$

Accordingly, the average error during the transient evolution is given by:

$$e_{av} = \frac{1}{t_{end} - t_{init}} \int_{t_{init}}^{t_{end}} e\,(t)\,\mathrm{d}t, \tag{3.15}$$

and the steady-state error corresponds to the one obtained for the last time step:

$$e_{ss} = e\,(t_{end})\,. \tag{3.16}$$

For the CGD case, the integration along time is replaced by an integration along the radius, from r_{init} to r_{final}.

3.5.1 Homogeneous flows

Three homogeneous flows were used to infer the network predictive capabilities. These are:

1. simple shear: $\left[0,\ 5x_3,\ 0\right]^{\mathrm{T}}$
2. shear-stretch: $\left[\frac{1}{10}x_1 + \frac{1}{10}x_2,\ \frac{1}{10}\,x_2,\ -\frac{1}{5}x_3\right]^{\mathrm{T}}$
3. shear-planar: $\left[\frac{3}{10}x_1 + \frac{1}{10}x_3,\ -\frac{3}{10}x_2,\ 0\right]^{\mathrm{T}}$

The $\mathbf{A}_2$ evolution equation was integrated in time up to $t = 20$ s, which is twice the training range, in order to also assess the neural network extrapolation capacity. Figure 3.7 shows the temporal evolution of A_{ij}. For the case of simple-shear, the flow and shearing directions are aligned with x_2 and

x_3, respectively. In shear flows, as time increases, the fiber aligns in the flow direction, x_2, thus A_{22} tends towards 1. The off-diagonal component, A_{23} initially starts from zero, and rises up to 0.42 s, indicating that between directions x_2 and x_3 lays one of the principal axes of orientation [21], and then evolves towards 0. The remaining off-diagonal components are zero throughout the entire time history.

For the shear-stretch case, the flow superimposes a biaxial extensional flow on the plane defined by x_1 and x_2, with shearing normal to x_1 and along x_2. The shear flow will orient the fibers along x_1, thus increasing A_{11}, whilst in the biaxial flow, due to symmetric flow conditions, will equally contribute to A_{11} and A_{22}. The combination of both, shows a predominant orientation along the x_1 direction as denoted by the quasi-monotonic increase of A_{11} with time.

Finally, the shear-planar flow, combines an extensional flow along x_1 with shearing normal to x_1 and along x_3. Both flows synergetically orient the fiber in the x_1 direction, thus increasing the contribution of the A_{11} component and subsequent reduction of the remainder components towards zero.

The qualitative (visual) analysis of the results shows a strong correspondence between the predicted and actual dynamics, as demonstrated by the overlapping curves. This is particularly noteworthy in the extrapolation region ($t > 10$), even for the shear-stretch flow, where the system has not yet reached a steady state.

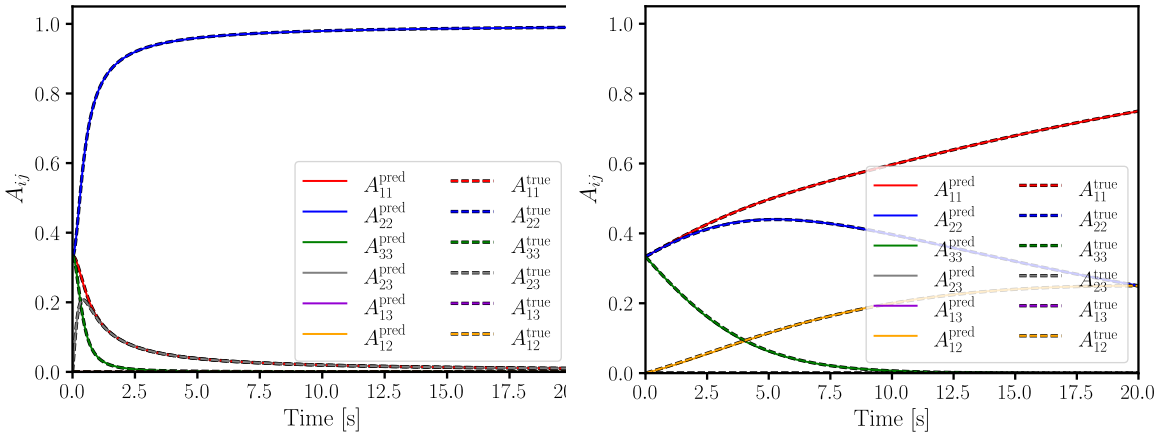
The error metrics for these flows are reported in Table 3.2. The quantitative results corroborate the qualitative analysis, where small errors are observed. The maximum of both steady and averaged error occurs for the shear-stretch flow, which is the only tested flow that does not reach steady state conditions.

Table 3.2: Error metrics for homogeneous flows

	simple-shear	shear-stretch	shear-planar
e_{av}	0.00053	0.00107	0.00074
e_{ss}	0.0003	0.00054	0.00026

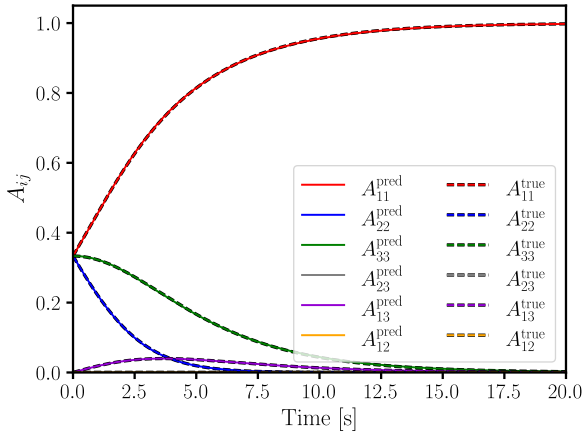
3.5.2 Combined flow

A more challenging test is posed when the state of orientation is not allowed to reach a steady condition. For this, three sequential homogeneous flows was used to describe the flow history. The series begins with simple-shear in the $x_1 - x_2$ direction, followed by a combination of shear in the $x_2 - x_3$ direction and stretching along x_3. The series ends with a pronounced stretching along x_1, while maintain the shear directions of the previous flow. The velocity gradients are given in Equation (3.17).



(a) simple-shear

(b) shear-stretch



(c) shear-planar

Figure 3.7: Model prediction for simple-shear(a), shear-stretch (b) and shear-planar(c).

$$L_{ij} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad 0 \leq t < 10 \tag{3.17a}$$

$$L_{ij} = \begin{bmatrix} -\frac{1}{20} & 0 & 0 \\ 0 & -\frac{1}{20} & 1 \\ 0 & 0 & \frac{1}{10} \end{bmatrix}, \qquad 10 \leq t < 20 \tag{3.17b}$$

$$L_{ij} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\frac{1}{2} & 1 \\ 0 & 0 & -\frac{1}{2} \end{bmatrix} \qquad 20 \leq t < 30 \tag{3.17c}$$

For comparison purposes Equation (2.4a) was integrated from isotropic state, using the IRD model, Equation (2.8), with a $C_I = 10^{-5}$ to minimize the effect of diffusion. Half of the unit sphere

was discretized, using $n_\theta = n_\phi = 256$ elements, with a power-law scheme [36] and a Courant number of 0.01.

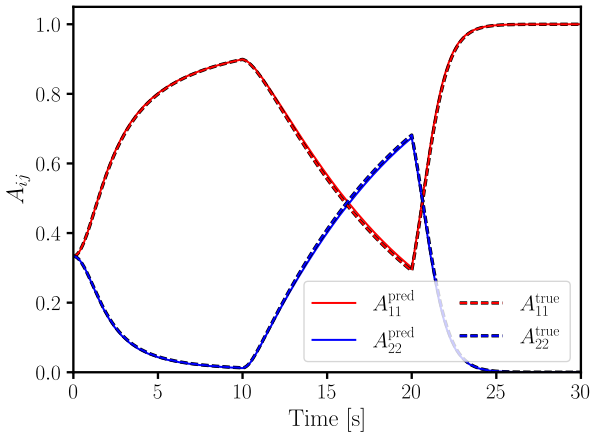


Figure 3.8: Evolution of A_{11} and A_{22} for the combined flow test case.

Analysis of Figure 3.8 shows again a fairly good agreement between the ODF integration and the UODE for the most relevant components of $\mathbf{A}_2$. A slight mismatch can be observed during the second phase of the flow history, which is reduced in the last phase due to its quick orientation towards the x_1 direction. The average and steady state error are, respectively, 0.00502 and 7×10^{-5}. The average error is almost 5 times higher than the error of the homogeneous shear-stretch flow reported in Table 3.2. However, the results show that the model can still accurately capture the dynamics of the sequential flow history.

3.5.3 Center-gated disk

For an isothermal, Newtonian fluid, the velocity field in a CGD is given by [9]:

$$\mathbf{v}\left(r,\theta,z\right) = \left[\frac{3Q}{8\pi r b}\left(1-\frac{z^2}{b^2}\right),\quad 0,\quad 0\right]^{\mathrm{T}}, \tag{3.18}$$

where Q is the volumetric flow-rate, r is the radial location, b is the disk half thickness, and $z = 0$ is the mid-plane. This parabolic velocity profile shows a maximum at the mid-plane and a minimum at the wall. The magnitude of velocity decreases along the radial direction. In local Cartesian coordinates, where (x_1, x_2, x_3), corresponds to (r, θ, z), the velocity gradient is given by [9]:

$$L_{ij} = \frac{3Q}{8\pi r b}\begin{bmatrix} -\frac{1}{r}\left(1-\frac{z^2}{b^2}\right) & 0 & -\frac{2}{b}\frac{z}{b} \\ 0 & \frac{1}{r}\left(1-\frac{z^2}{b^2}\right) & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.19}$$

Analysis of the velocity gradient reveals that a fiber moving along the radius experiences a combination of shear and planar flows. At the mid-plane $(z = 0)$, pure planar flow is obtained and at

the wall ($z = b$), a dominant shear flow is imposed. The ratio between the shear and stretch rates increases from the mid-plane towards the wall, and decreases along the radius.

An equivalent representation for the velocity gradient is given by Altan and Rao [37], as:

$$L_{ij}\left(\bar{x}_1, \bar{x}_3\right) = \begin{bmatrix} -\frac{\left(1-\bar{x}_3^2\right)}{\bar{x}_1^2} & 0 & -\frac{2\bar{x}_3}{\bar{x}_1} \\ 0 & \frac{\left(1-\bar{x}_3^2\right)}{\bar{x}_1^2} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{3.20}$$

where $\bar{x}_i = x_i/b$. Using this coordinates transformation, the authors derived a solution for a center-gated disk in the diluted state, based on the inverse of the usually denominated rotation tensor, $\mathbf{E}^{-1}$, given by:

$$\mathbf{E}^{-1} = \begin{bmatrix} \bar{x}_1 & 0 & \frac{\bar{x}_3\left(1-\bar{x}_1^2\right)}{\bar{x}_3^2-1} \\ 0 & \frac{1}{\bar{x}_1} & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.21}$$

In the current notation: $\mathbf{E}^{-\mathrm{T}} \cdot \mathbf{E}^{-1}$ is equal to $\mathbf{B}^{-1}$, where the superscript $^{-\mathrm{T}}$ indicates the transpose operation of $\mathbf{E}^{-1}$.

The true dynamics for the CGD where generated through the usage of Equation (3.7), where 9 linearly spaced points from $\bar{x}_3 = 0.1$ to $\bar{x}_3 = 0.9$ where used to compute the evolution of A_{ij} as a function of the normalized radius, from $\bar{x}_1 = 1$ up to $\bar{x}_1 = 20$.

The predicted values where computed with $Q = 1$, $b = 1$, and $z = \bar{x}_3$. The evolution equation of $\mathbf{A}_2$ as a function of the radius was computed by $\frac{\mathrm{d}\mathbf{A}_2}{\mathrm{d}r} = \frac{1}{u_r}\frac{\mathrm{d}\mathbf{A}_2}{\mathrm{d}t}$, using the same solver settings as defined during the training of the network.

Figure 3.9 shows the evolution of A_{11} and A_{22} as a function of $\bar{x}_1$. Analysis of the Figure shows that near the mid-plane, where there is a small shear contribution, the flow is dominated by planar extension in the θ-direction (transverse to $\bar{x}_1$), thus showing larger values for A_{22}. With increasing distance from mid-plane, the effect of shear increases up to a point where the fibers are mostly oriented along the main flow direction, thus increasing A_{11}. The microstructure for this case appears to be fully developed for $\bar{x}_1 > 10$ [7, 37].

Table 3.3: Error metrics calculated along the normalized disk thickness ($\bar{x}_3$) for the center-gated disk case

$\bar{x}_3$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
e_{av}	0.00051	0.00069	0.00069	0.00129	0.0032	0.00663	0.00207	0.0014	0.00162
e_{ss}	0.0004	0.00012	0.00038	0.00187	0.00453	0.00858	0.00247	0.00166	0.00175

Table 3.3 contains the error metrics of the CGD case. The largest errors for both average and steady-steady errors is obtained at $\bar{x}_3 = 0.6$. The overall predictive capability of the UODE matches

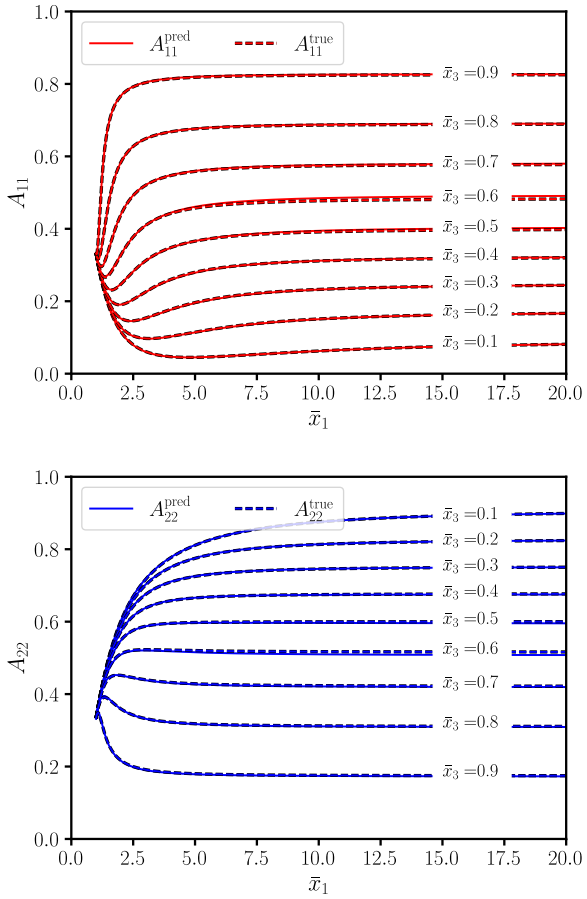


Figure 3.9: Evolution of the main $\mathbf{A}_2$ tensor components along the normalized radius ($\bar{x}_1$) at different $\bar{x}_3$ locations, for the center gated-disk case: (a) A_{11} and (b) A_{22}.

fairly well with the real system dynamics.

3.5.4 Preliminary analysis in the presence of fiber interactions

Given that the proposed UODE closure was trained using data in the dilute state, $C_I = 0$, in this section a preliminary study is presented to show the potential of the proposed approach when compared with state-of-the-art closure relations used in proprietary applications, namely the IBOF [9] and ORE [8], together with the neural closure, NNORT [10], described in Section 3.2. To generate the results, the second-order tensor evolution equation was integrated using the IRD model with $C_I = 10^{-2}$ (a common value used in the literature [7, 9, 10] that highlights the effect the tested flow on fiber orientation dynamics), under simple-shear $\mathbf{v} = \begin{bmatrix} x_2, & 0, & 0 \end{bmatrix}^{\mathrm{T}}$, from $t = 0$ s up to $t = 40$ s, starting from the isotropic state. The integration settings are the same as those used in Subsection 3.5.2.

Figure 3.10 shows the time evolution of the A_{11} and A_{12} components. The integration of the ODF, Equation (2.4a), represents the true system dynamics. The accuracy of the state-of-the-art closure relations and the proposed UODE is fairly similar, with the NNORT closure presenting the

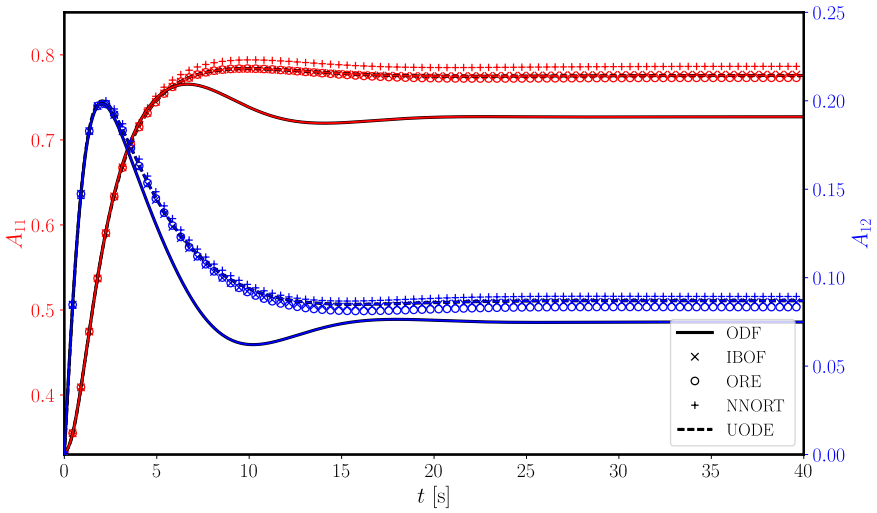


Figure 3.10: Comparison of the temporal evolution of A_{11} and A_{12} in simple-shear.

least agreement. This lack of accuracy in simple-shear has also been observed in other works.

All the analyzed systems have a similar dynamic, with the A_{11} component increasing up to $t \approx 9.8$ s, followed by a small decrease and plateauing at $A_{11} \approx 0.775$. The A_{12} off-diagonal component, rises quickly from 0 to 2 s, and afterwards decreases to a steady-state value. During the initial transient, all closures match well the true system dynamics. However, as the effect of diffusion becomes more prominent, none of the closure relations are able to adequately replicate the correct evolution. The ODF results show a smaller value of steady-state for both A_{11} and A_{12}, with a distinct temporal evolution. In practice, when using standard closure relationships, it is common to "calibrate" the model by adjusting the value of C_I to match the steady-state orientation value [21], however, the transient region becomes less accurate. Given the relatively good relationships with the data in a dilute state obtained with the proposed approach, the authors are currently working on incorporating the effect of diffusion into UODE to infer its contribution to the closure problem. This will enable to address problems that are increasingly relevant in practice. However, based on the preliminary studies conducted, it is evident that extending the proposed approach is not a simple task and will require a comprehensive investigation to be developed.

3.6 Conclusion

In this work, a universal ordinary differential equation approach was used to infer a data-driven closure relationship for fibers in the dilute state.

To restrict the search space of the problem, admissible bounds for an orthotropic correction to the linear closure in eigenspace was determined based on symbolic computation. The independent eigenvalues of $\mathbf{A}_2$ and the bounded output of the network were used to obtain admissible correction values.

The model was trained with data for different incompressible flows, which were synthetically generated by solving an elliptical integral after solving an evolution equation for the second order Piola tensor, over a certain time range.

Three test cases were used to assess the model performance. The first test, used homogeneous flows that were not included in the training pool of the UODE and was integrated in a range twice as large as the one used for training. The second test, used a sequential set of combined flows, in which steady-state conditions were just achieved in the last flow. Finally, a test with a non-homogeneous flow in a center gated disk was conducted, where different combinations of shear and extensional flow conditions were assessed.

All the results obtained show a good agreement between the dynamics predicted by the UODE and the true dynamics of fiber orientation, evidencing the clear benefits of the developed approach. The results obtained for pure shear flow with diffusion using the NNORT [10] closure relation, were worse compared to the ones obtained from the approach proposed in this work, which only considers hydrodynamic effects. This highlights the need for a systematic and careful approach to include diffusion in data-driven models.

Appendices

Appendix A Mandel Notation

$\tilde{\mathbb{A}}$ is represented as a 6×6 matrix. The corresponding entries to A_{ijkl} are [19]:

$$\tilde{\mathbb{A}} = \begin{bmatrix} A_{1111} & A_{1122} & A_{1133} & \sqrt{2}A_{1123} & \sqrt{2}A_{1113} & \sqrt{2}A_{1112} \\ A_{1122} & A_{2222} & A_{2233} & \sqrt{2}A_{2223} & \sqrt{2}A_{2213} & \sqrt{2}A_{2212} \\ A_{1133} & A_{2233} & A_{3333} & \sqrt{2}A_{3323} & \sqrt{2}A_{3313} & \sqrt{2}A_{3312} \\ \sqrt{2}A_{1123} & \sqrt{2}A_{2223} & \sqrt{2}A_{3323} & 2A_{2233} & 2A_{3312} & 2A_{2213} \\ \sqrt{2}A_{1113} & \sqrt{2}A_{2213} & \sqrt{2}A_{3313} & 2A_{3312} & 2A_{1133} & 2A_{1123} \\ \sqrt{2}A_{1112} & \sqrt{2}A_{2212} & \sqrt{2}A_{3312} & 2A_{2213} & 2A_{1123} & 2A_{1122} \end{bmatrix}. \tag{A.1}$$

Appendix B Parameterized bounds

The bound equations used in this work are:

$$d_1 \geq -\frac{\lambda_1}{7} - \frac{\lambda_2}{7} + \frac{1}{35} \quad , \tag{B.1a}$$

$$d_2 \geq \frac{\lambda_2}{7} - \frac{4}{35} \quad , \tag{B.1b}$$

$$d_3 \geq \frac{\lambda_1}{7} - \frac{4}{35} \quad . \tag{B.1c}$$

$$d_1 < \frac{1}{a}\left(-5\lambda_1^2 + 25\lambda_1\lambda_2 + \lambda_1 - 5\lambda_2^2 + \lambda_2\right) \quad , \tag{B.2a}$$

$$\begin{aligned} d_2 < \frac{1}{b}\big(&-1225d_1\lambda_1^2 - 2450d_1\lambda_1\lambda_2 + 1225d_1\lambda_1 - 1225d_1\lambda_2^2 \\ &+ 1400d_1\lambda_2 - 140d_1 - 175\lambda_1^3 + 700\lambda_1^2\lambda_2 + 210\lambda_1^2 \\ &+700\lambda_1\lambda_2^2 - 780\lambda_1\lambda_2 - 55\lambda_1 - 80\lambda_2^2 + 80\lambda_2 + 4\big) \end{aligned} \quad , \tag{B.2b}$$

$$\begin{aligned} d_3 < \frac{1}{c}\big(&-1225d_1d_2 - 1225d_1\lambda_1^2 - 2450d_1\lambda_1\lambda_2 + 1400d_1\lambda_1 \\ &- 1225d_1\lambda_2^2 + 1400d_1\lambda_2 - 280d_1 - 1225d_2\lambda_2^2 \\ &+ 1050d_2\lambda_2 - 105d_2 + 700\lambda_1^2\lambda_2 - 80\lambda_1^2 + 700\lambda_1\lambda_2^2 \\ &-780\lambda_1\lambda_2 + 80\lambda_1 - 80\lambda_2^2 + 80\lambda_2 - 8\big) \end{aligned} \quad , \tag{B.2c}$$

$$a = 35\,(\lambda_1 + \lambda_2), \tag{B.2d}$$

$$b = 35\left(35d_1 + 5\lambda_1 + 35\lambda_2^2 - 30\lambda_2 - 1\right), \tag{B.2e}$$

$$c = 35\left(35d_1 + 35d_2 + 35\lambda_1^2 - 30\lambda_1 + 3\right). \tag{B.2f}$$

Appendix C Availability of data and material

The data that support the findings of this study are openly available at `https://github.com/Computational-Rheology/data-driven-closure`.

References

[1] Babatunde O. Agboola, David A. Jack, and Stephen Montgomery-Smith. Effectiveness of recent fiber-interaction diffusion models for orientation and the part stiffness predictions in injection molded short-fiber reinforced composites. *Composites Part A: Applied Science and Manufacturing*, 43(11):1959–1970, 2012.

[2] David A. Jack and Douglas E. Smith. Elastic properties of short-fiber polymer composites, derivation and demonstration of analytical forms for expectation and variance from orientation tensors. *Journal of Composite Materials*, 42(3):277–308, 2008.

[3] Delphine Dray, Pierre Gilormini, and Gilles Régnier. Comparison of several closure approximations for evaluating the thermoelastic properties of an injection molded short-fiber composite. *Composites Science and Technology*, 67(7):1601–1610, 2007.

[4] Adrien Scheuer, Amine Ammar, Emmanuelle Abisset-Chavanne, Elias Elías Elias Cueto, Adrien Scheuer, Amine Ammar, Emmanuelle Abisset-Chavanne, Elias Elías Elias Cueto, Francisco Chinesta, Roland Keunings, and Suresh G. Advani. Data-driven upscaling of orientation kinematics in suspensions of rigid fibres. *Computer Modeling in Engineering and Sciences*, 117(3):367–386, 2018.

[5] Suresh G. Advani and Charles L. Tucker. The use of tensors to describe and predict fiber orientation in short fiber composites. *Journal of Rheology*, 31(8):751–784, 1987.

[6] Suresh G. Advani and Charles L. Tucker. Closure approximations for three□dimensional structure tensors. *Journal of Rheology*, 34(3):367–386, 1990.

[7] Joaquim S. Cintra and Charles L. Tucker. Orthotropic closure approximations for flow□induced fiber orientation. *Journal of Rheology*, 39(6):1095–1122, 1995.

[8] Brent Eugene VerWeyst. *Numerical Predictions of Flow-Induced Fiber Orientation in Three-Dimensional Geometries*. PhD thesis, University of Illinois at Urbana-Champaign, 1998.

[9] Du Hwan Chung and Tai Hun Kwon. Invariant-based optimal fitting closure approximation for the numerical prediction of flow-induced fiber orientation. *Journal of Rheology*, 46(1):169–194, 2002.

[10] Najam ul Qadir and David A. Jack. Modeling fibre orientation in short fibre suspensions using the neural network-based orthotropic closure. *Composites Part A: Applied Science and Manufacturing*, 40(10):1524–1533, 2009.

[11] David A. Jack, Bryan Schache, and Douglas E. Smith. Neural network-based closure for modeling short-fiber suspensions. *Polymer Composites*, 31(7):1125–1141, 2010.

[12] Kevin Breuer, Markus Stommel, and Wolfgang Korte. Analysis and evaluation of fiber orientation reconstruction methods. *Journal of Composites Science*, 3(3), 2019.

[13] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations, 2019.

[14] Idelfonso B. R. Nogueira, Vinicius V. Santana, Ana M. Ribeiro, and Alírio E. Rodrigues. Using scientific machine learning to develop universal differential equation for multicomponent adsorption separation systems. *The Canadian Journal of Chemical Engineering*, 100(9):2279–2290, 2022.

[15] Albin Heimerson and Johan Ruuskanen. Extending microservice model validity using universal differential equations. *IFAC-PapersOnLine*, 56(2):2401–2406, 2023. 22nd IFAC World Congress.

[16] Zhilu Lai, Charilaos Mylonas, Satish Nagarajaiah, and Eleni Chatzi. Structural identification with physics-informed neural ordinary differential equations. *Journal of Sound and Vibration*, 508:116196, 2021.

[17] Mattia Silvestri, Federico Baldo, Eleonora Misino, and Michele Lombardi. An analysis of universal differential equations for data-driven discovery of ordinary differential equations. In Jiří Mikyška, Clélia de Mulatier, Maciej Paszynski, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M.A. Sloot, editors, *Computational Science – ICCS 2023*, pages 353–366, Cham, 2023. Springer Nature Switzerland.

[18] Fernando Arrais R.D. Lima, Carine M. Rebello, Erbet A. Costa, Vinícius V. Santana, Marcellus G.F. de Moares, Amaro G. Barreto, Argimiro R. Secchi, Maurício B. de Souza, and Idelfonso B.R. Nogueira. Improved modeling of crystallization processes by universal differential equations. *Chemical Engineering Research and Design*, 200:538–549, 2023.

[19] Julian Karl Bauer and Thomas Böhlke. Variety of fiber orientation tensors. *Mathematics and Mechanics of Solids*, 27(7):1185–1211, 2022.

[20] Fransisco Folgar and Charles L. Tucker. Orientation Behavior of Fibers in Concentrated Suspensions. *Journal of Reinforced Plastics and Composites*, 3(2):98–119, 1984.

[21] C.L. Tucker. *Fundamentals of Fiber Orientation: Description, Measurement and Prediction*. Hanser Publishers, Munich, Germany, 2022.

[22] Edward B Saff and Amo BJ Kuijlaars. Distributing many points on a sphere. *The mathematical intelligencer*, 19:5–11, 1997.

[23] Scipy. solve_ivp routine. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html`. Accessed: 2024-01-10.

[24] Steven M. Dinh and Robert C. Armstrong. A Rheological Equation of State for Semiconcentrated Fiber Suspensions. *Journal of Rheology*, 28(3):207–227, 06 1984.

[25] Stephen Montgomery-Smith, Wei He, David A. Jack, and Douglas E. Smith. Exact tensor closures for the three-dimensional jeffery's equation. *Journal of Fluid Mechanics*, 680:321–335, 2011.

[26] *mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 1.3.0)*, 2023. `http://mpmath.org/`.

[27] Jennifer Brucker, Wolfgang G Bessler, and Rainer Gasper. Grey-box modelling of lithium-ion batteries using neural ordinary differential equations. *Energy Inform.*, 4(S3), September 2021.

[28] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[29] Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

[30] Patrick Kidger and Cristian Garcia. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming workshop at Neural Information Processing Systems 2021*, 2021.

[31] DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020.

[32] Christer Ericson. *Real-Time Collision Detection*. CRC Press, Inc., USA, 2004.

[33] George T. Gilbert. Positive definite matrices and sylvester's criterion. *The American Mathematical Monthly*, 98(1):44–46, 1991.

[34] Wolfram Research, Inc. Mathematica, Version 14.0. Champaign, IL, 2024.

[35] Constantin Krauß, Julian Karl Bauer, Johannes Mitsch, Thomas Böhlke, and Luise Kärger. On the averaging and closure of fiber orientation tensors in virtual process chains. *Journal of Elasticity*, 2024.

[36] J. Férec, M. Heniche, M.C. Heuzey, G. Ausias, and P.J. Carreau. Numerical solution of the fokker–planck equation for fiber suspensions: Application to the folgar–tucker–lipscomb model. *Journal of Non-Newtonian Fluid Mechanics*, 155(1):20–29, 2008.

[37] M. Cengiz Altan and Bharath N. Rao. Closed□form solution for the orientation field in a center□gated disk. *Journal of Rheology*, 39(3):581–599, 05 1995.

Chapter 4 Non-Orthotropic Contributions in Flow-Induced Fiber Orientation: Toward Richer Closure Models

Abstract:

Fiber orientation modeling of discontinuous fiber-reinforced polymers plays a crucial rule for accurately estimating the mechanical and rheological properties of manufactured parts. Traditional closure models for the evolution of the second-order orientation tensor typically assume an orthotropic symmetry when expressed in the principal axes of orientation, which can limit their accuracy in flows where significant fiber interactions exist.

In this work, we investigate the development of non-orthotropic components in flow-induced fourth-order orientation tensors for a range of homogeneous flows and its impact in the orientation dynamics. Synthetic data were generated via numerical solution of the Folgar-Tucker model and analyzed within the harmonic decomposition framework.

Our results show that while orthotropic symmetry yields a good approximation at lower levels of diffusion, at higher ones and for flows like simple-shear, the orthotropic restriction should be relaxed. To highlight this, a polynomial closure model incorporating all deviatoric components was developed and shown to more accurately capture the orientation dynamics than pure orthotropic models. Our findings suggest that relaxing the orthotropic restriction and including additional objective flow kinematic information can be used as a potential venue for richer three-dimensional closure relations for use in flow induced fiber orientation modeling.

Keywords: Fiber orientation modeling; Flow-induced orientation; Non-orthotropic contributions

This chapter was adapted from B. Ramoa, J.M. Nóbrega, and F. Chinesta, Non-Orthotropic Contributions in Flow-Induced Fiber Orientation: Toward Richer Closure Models, submitted to the journal Modelling and Simulation in Materials Science and Engineering (2025)

4.1 Introduction

Discontinuous fiber-reinforced polymers are widely used for manufacturing technical components. The relevant mechanical and rheological properties of these materials depend the microstructure that is developed during processing, which is partially governed by the concentration and spacial arrangement of the reinforcing phase.

Accurately predicting the final state of orientation enables more reliable estimates of a part's physical properties. Over the past 40 years, numerous models have been proposed to capture the dynamics of discontinuous fibers [1–7], relying on the use of orientation tensors as a macro-descriptor of the state of orientation. Because the evolution equation for these quantities involves the next even-order tensor, a closure approximation is required using information from the lower-order tensor [1]. Consequently, a variety of closure models have been reported in the literature [8–13] with varying degrees of accuracy.

In the absence of diffusion, the second-order orientation tensor, has a one-to-one correspondence with the fourth-order counterpart [9, 11, 14]. Assuming an initial isotropic state, $\mathbf{A}$ and $\mathbb{A}$ can be calculated from a relationship with the Piola tensor, by computing elliptic integrals [15]. This is possible because the principal axes of $\mathbf{B}^{-1}$ coincide with those of $\mathbf{A}$ and the resulting fourth-order tensor will be orthotropic in the coordinate system defined by the principal directions of $\mathbf{A}$ [11].

When diffusion is present in the system, closures enforcing orthotropy fail to accurately capture specific flow-induced orientation phenomena, such as the orientation peak formed under simple-shear conditions [9, 10].

The traditional fitted closures [9, 10, 16] enforce the orthotropic character by taking all directional information from the principal directions of $\mathbb{A}$. However, it is known from the work of Cintra and Tucker [10] that flow-induced orientation can produce fourth-order orientation tensors with non-orthotropic components, and that the orthotropy restriction can be relaxed if $\mathbb{A}$ is allowed to depend on other tensors in addition to $\mathbf{A}$. Building on the work of Hand [17], Bay [18] proposed a general closure formulation dependent on both A_{ij} and the rate-of-deformation tensor, D_{ij}. However, this family of closures did not gain traction due to the assumed lack of usefulness when predicting mechanical properties [10]. Nonetheless, as noted in the work of Tucker [13], an improvement in the accuracy of mechanical properties can be obtained by considering richer information then the one provided by the orthotropic assumption. In fact, the author initiated a new functional family of non-orthotropic planar closures by considering A_{ij} and the angle between the principal direction of A_{ij} and D_{ij}, yielding significant improvements in the prediction of orientation dynamics. This formulation, however, is restricted to planar flows, and a fully three-dimensional closure yet to be realized.

In this study, we investigate the non-orthotropic character of flow-induced fourth-order orientation tensors generated using the scalar rotary-diffusivity model of Folgar and Tucker [19]. The analysis is

carried out within the decomposition framework proposed by Bauer and Böhlke [20] and focuses on the influence of non-orthotropic components on the evolution of the second-order orientation tensor.

The results demonstrate that the orthotropic approximation is a viable option at low levels of diffusion, as the values of non-orthotropic components remain limited. In a higher diffusion regime, however, these contributions become significant and should not be neglected, as they substantially enhance the accuracy of the resulting closure approximations.

The remaining paper is organized as follows. Section 4.2 details the procedure used to generate synthetic data. Section 4.3 analyzes the appearance of non-orthotropic components in the flow-induced orientation of selected flows and Section 4.4 highlights how the orientation dynamics is affected by the non-orthotropic contributions in simple-shear flow. Finally, Section 4.5 summarizes the main conclusions of this work.

4.2 Data generation

Synthetic data for flow-induced orientation in the presence of isotropic rotary diffusion can be generated though numerical integration of Equation (2.4) or via an ensemble average of a stochastic differential equation where a Wiener process is added to Jeffery's equation [21]. The choice between the two approaches depends on the computational efficiency required for a given level of C_I and the flow conditions (turbulent or laminar). The solution of the stochastic equation constitutes an embarrassingly parallel problem whose numerical solution scales very efficiently with the available computational resources.

In this work, data was generated through numerical solution of Equation (2.4) using the isotropic rotary diffusion model, following the methodology described in [11], with the numerical routine available in [22] adapted to Julia programming language [23]. The computational domain covered one hemisphere of the unit sphere and was discretized using 256 elements in both the θ and ϕ directions. The numerical solution uses an implicit scheme with periodic boundary conditions [22]. An Euler discretization is used for the transient term and central differences for the spatial terms. The time step was determined based on a grid Courant number of $Co = 0.05$. An isotropic distribution ($\psi = 1/4\pi$) was used as initial condition, and the evolution equation solved over the time interval $[0, 40]$ s. Two diffusion levels where considered: a moderate level with $C_I = 1 \times 10^{-3}$, and a higher level with $C_I = 1 \times 10^{-2}$.

As a post-processing step, at each stored time step, the fourth-order orientation tensor was rotated to the principal frame of orientation, $\bar{\mathbb{A}}$, and the deviatoric component $\bar{\mathbb{A}}^{\mathrm{dev4}}$ computed by subtracting $\bar{\mathbb{A}}^{\mathrm{ISO}}$ and $\bar{\mathbb{A}}^{\mathrm{dev2}}$ from $\bar{\mathbb{A}}$.

For the sake of completeness, the data generation process using the stochastic method is reported in Appendix A using a stochastic method and the sampling procedure proposed by Saff and Kuijlaars [24].

4.3 Analysis of deviatoric components

The analysis of the deviatoric components, requires a rotation to the principal frame of orientation using the rotation matrix $\mathbf{R}$, whose columns are given by the principal directions of $\mathbf{A}$. However, this matrix is not unique, as a sign change in the of one of the eigenvectors yields another valid eigenvector of $\mathbf{A}$. Our observations show that this ambiguity influences both the sign and smoothness of the non-orthotropic components that develop during flow-induced orientation.

For processing the data, and in line with the approach of [13], we consider the angles between the principal directions of $\mathbf{A}$ and $\mathbf{D}$ and apply the same descending sorting of the eigenvalues to both tensors. At each time step, we compute the dot product of $\pm e_i^{\mathbf{D}}$ with $e_j^{\mathbf{I}}$ and $\pm e_i^{\mathbf{A}}$ with $e_j^{\mathbf{D}}$, then determine the sign of both objective tensors based on the highest dot product.

Figure 4.1 shows the trajectory of each considered flow with $C_I = 1 \times 10^{-2}$ in the *UBT* triangle, see Section 3.2, and will be referenced in a later discussion.

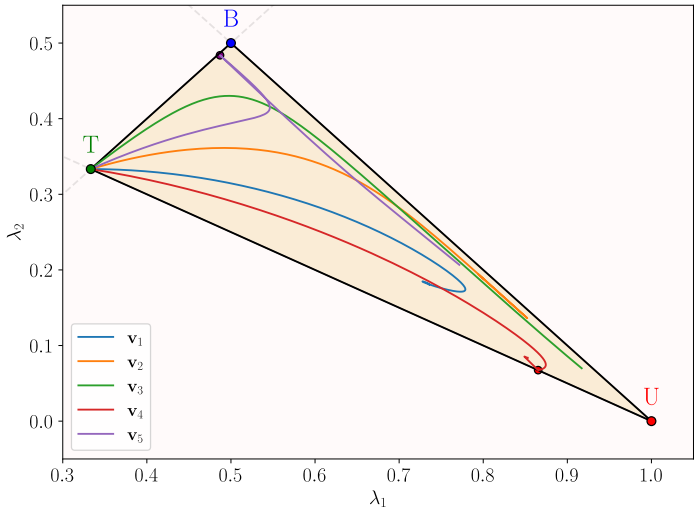


Figure 4.1: Trajectory of the different flows ($\mathbf{v}_1$ to $\mathbf{v}_5$) inside the UBT triangle for $C_I = 1 \times 10^{-2}$

4.3.1 Simple-shear

Simple-shear flow is a representative type of isochoric flow commonly encountered in a range of practical applications, particularly in the rheological characterization of fluids. In this subsection we examine the flow-induced orientation of a fiber under simple-shear flow with a velocity field defined by: $\mathbf{v}_1 = \begin{bmatrix} \dot{\gamma} x_3, & 0, & 0 \end{bmatrix}^{\mathrm{T}}$, with $\dot{\gamma} = 1$.

Under the considered profile, a material element moves along direction x_1 with a velocity proportional to its position along the x_3-direction. Decomposing the velocity gradient into its symmetric and anti-symmetric parts produces a non-null vorticity tensor, consequently this flow induces rigid-body rotation.

Figure 4.2a illustrates the transient evolution of the A_{ij} components starting from isotropic state. For this flow the relevant components are A_{11}, A_{33} and A_{13}, with the remaining off-diagonal components being null. The resulting flow-induced fiber orientation for this type of flow is known to align the fibers along the flow direction [11].

In the dilute regime, no mechanism exists to prevent a fiber from achieving full alignment, so A_{11} increases monotonically with time, approaching 1 asymptotically. Conversely, A_{33} decreases towards zero, while A_{22} has complementary dynamics due to the normalization condition. The only non-zero off-diagonal term, A_{13}, starts at zero, increases to a maximum at ≈ 2 s, and decays towards zero with increasing flow history, indicating that the principal axes of orientation lies between the $x_1 - x_3$ plane [11].

Analyzing the evolution of the d_i components in Figure 4.2b for the dilute regime shows that only the orthotropic trio ($d_{1\ldots3}$) has a significant contribution to $\bar{\mathbb{A}}$. The transient behavior of these components exhibits rather monotonic features, with d_1 and d_2 steadily decreasing over time, whilst d_3 increases throughout the flow history.

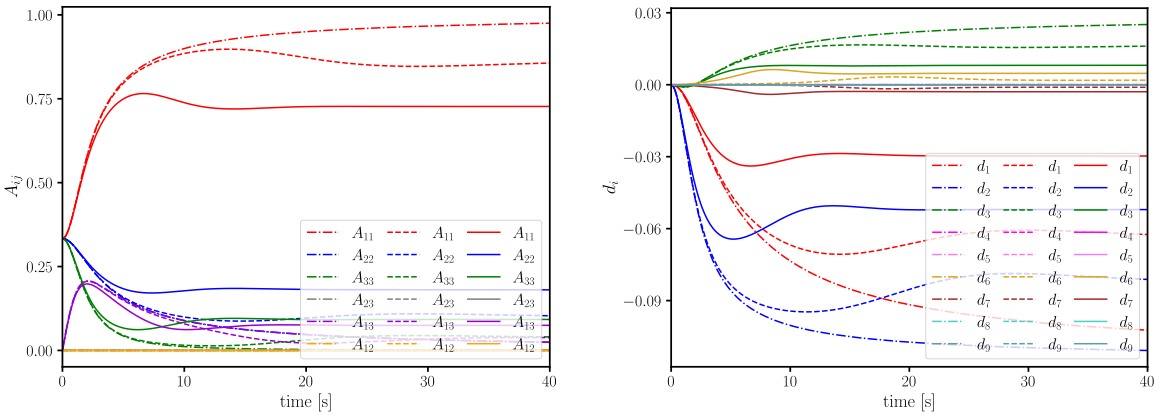
When diffusion is introduced into the system, the evolution of the A_{ij} components exhibits more complex features. At a moderate level of diffusion, the A_{11} component initially shows a similar dynamic behavior to the one observed in the dilute state. After some exposure to the flow, a shift in the dynamics occurs, preventing the full alignment along the flow direction. The A_{11} component peaks at $t \approx 14$s, followed by a decrease in the degree of alignment until a quasi-steady value is reached at the end of the flow history. This behavior becomes more pronounced at the highest level of diffusion where A_{11} has an overshoot at $t \approx 7$s before decreasing towards a steady-state value.

The evolution of the d_i components also show more complex dynamics. With increasing level of diffusion, the values of d_1 and d_2 form a valley during the development of the orientation peak, followed by a steady increase until reaching a plateau when the orientation reaches a steady state. The evolution of d_3 is more monotonic; as the level of diffusion increases this parameter reaches a steady-state faster. In line with what was reported by Cintra and Tucker [10], the fourth-order tensor has non-orthotropic components that develop in simple-shear. In the framework of the harmonic decomposition, this flow produces the non-orthotropic pair (d_6, d_7), which emerges earlier and with higher contribution as diffusion increases.

Describing the full fourth-order tensor is not a requirement to solve the evolution equation of $\mathbf{A}$ as only the result of the contraction $\mathbb{A} : \mathbf{D}$ is contributing to the dynamics. The result of this operation is given, in Mandel notation, by Equation (4.1), where only 6 components of $\mathbb{A}$ remain. However, the orientation dynamics shows that A_{1123} and A_{3312} are zero throughout the flow history.

$$\mathbb{A} : \mathbf{D} = \dot{\gamma} \begin{bmatrix} A_{1131} & A_{2231} & A_{3331} & \sqrt{2}A_{3312} & \sqrt{2}A_{3311} & \sqrt{2}A_{1123} \end{bmatrix}^{\mathrm{T}} \tag{4.1}$$

To highlight the contribution of each component of $\mathbb{A}$, the result of the contraction was



(a) Time evolution of A_{ij} components at different levels of C_I

(b) Time evolution of d_i components at different levels of C_I

Figure 4.2: Temporal evolution of A_{ij} and d_i components under the homogeneous velocity field $\mathbf{v}_1$. Dash-dot line $\longrightarrow C_I = 0$, dashed line $\longrightarrow C_I = 1 \times 10^{-3}$, solid line $\longrightarrow C_I = 1 \times 10^{-2}$

decomposed into linear $(C, \lambda_1, \lambda_2)$ and orthotropic $(d_{1..3})$ contributions and non-orthotropic (d_{4-5}, d_{6-7} and d_{8-9}) pair contributions, for the highest level of diffusion considered in this case study, as illustrated in Figure 4.3. The secondary axis illustrates the relative contribution of each term to the overall value of each non-negligible term in the contraction. This contribution was computed following equation (4.2).

$$\Delta = 100 \frac{|x_i|}{\sum_{i=0} |x_i|} \tag{4.2}$$

where, x_i is the component in analysis, e.g., linear contribution with orthotropic or any pair of non-orthotropic components.

Analysis of the figure shows that the terms contributing to the dynamics of A_{11} are mainly driven by the linear-orthotropic terms, since they contribute to the bulk of A_{1131}.

For the A_{13} and A_{33} components, the main contributors are again the linear-orthotropic portion. However, for the latter component, the contribution of the $d_6 - d_7$ pair is more pronounced, specially in the formation of the orientation peak and at steady-state.

Except for the initial isotropic state, the second principal direction of $\mathbf{A}$ remains aligned with the second principal direction of $\mathbf{D}$. The evolution of principal directions corresponds to a rotation within the $\mathbf{e}_1^{\mathrm{A}} - \mathbf{e}_3^{\mathrm{A}}$ plane as depicted in Figure 4.4.

4.3.2 Biaxial flow with shear

Elongational/stretch flows are shear-free flows, characterized by a velocity gradient with no off-diagonal components. Under the assumption of incompressibility, these flows produce a vorticity

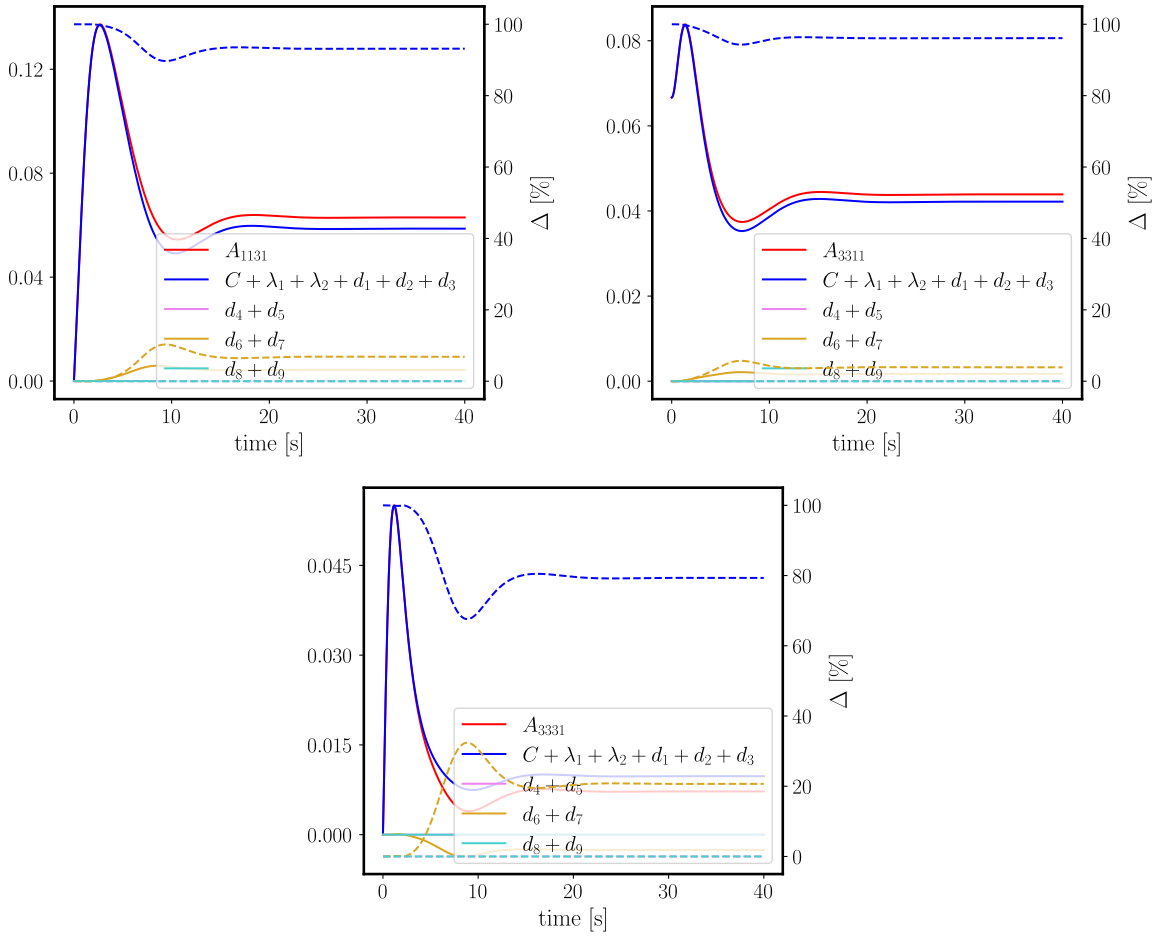


Figure 4.3: Split of active components of $\mathbb{A} : \mathbf{D}$ in Canonical space for flow $\mathbf{v}_1$ at $C_I = 1 \times 10^{-2}$. Dashed lines correspond to the relative contribution of each component.

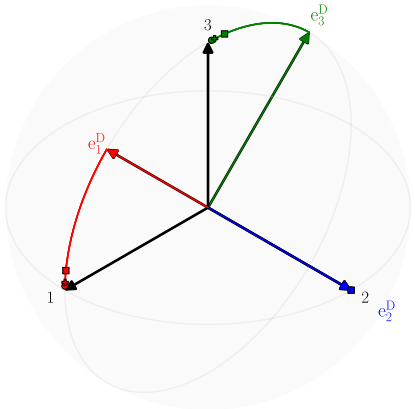


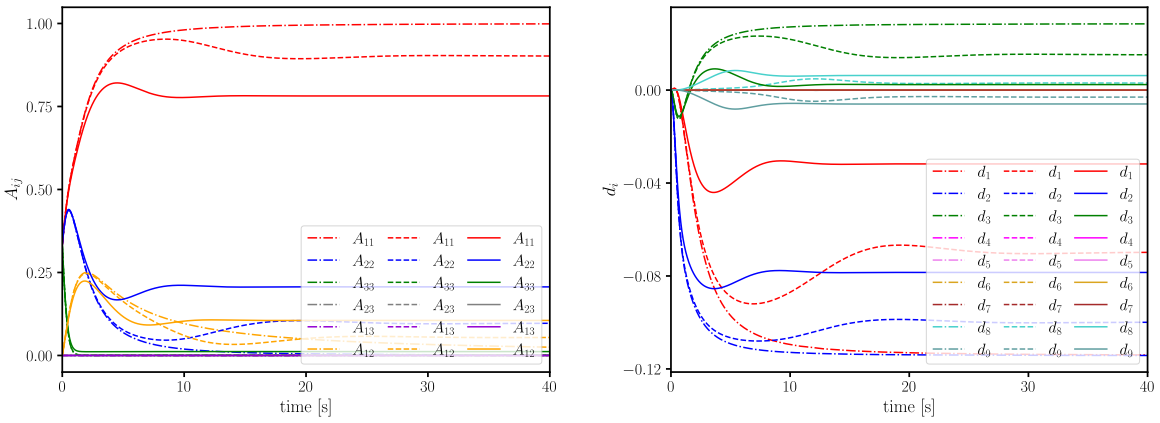
Figure 4.4: Trajectory of principal directions of $\mathbf{A}$ for flow $\mathbf{v}_1$. The principal directions $\mathbf{e}_1^A$, $\mathbf{e}_2^A$, and $\mathbf{e}_3^A$ are represented in red, blue, and green, respectively. The final position along each trajectory are marked with a circle, cross, and square for $C_I = 0$, $C_I = 1 \times 10^{-3}$, and $C_I = 1 \times 10^{-2}$, respectively.

tensor equal to zero and a rate-of-deformation tensor with null trace.

In such flows the material is stretched uniformly along one or two directions while contracting in the remaining. These flows are known to align fibers in the stretching direction [11]. Introducing a shear component in the velocity profile introduces a non-trivial coupling between rotation and stretching, which can lead to complex orientation trajectories.

In this section, we analyze the flow-induced orientation under the velocity field: $\mathbf{v}_2 = \left[\dot{\epsilon}x_1 + \dot{\gamma}x_2, \quad \dot{\epsilon}x_2, \quad -2\dot{\epsilon}x_3\right]^{\mathrm{T}}$, with $\dot{\gamma}$ and $\dot{\epsilon}$ set to unitary values. The biaxial flow component should uniformly stretch the fiber within the $x_1 - x_2$ plane and contract along the x_3 direction. The shear component induces a shearing motion in the $x_1 - x_2$ plane.

The evolutions of the A_{ij} and d_i components for this flow are depicted in Figure 4.5. The combined effect of both flows, results in a synergistic contribution along the x_1 direction. The only off-diagonal component that evolves during the flow history is the A_{12} component, indicating that the principal axes of orientation lies between the $x_1 - x_2$ plane. As the level of diffusion increases, a similar behavior to that reported for simple-shear can be observed. The overall degree of orientation decreases due to the diffusion contribution. After some exposure to the flow, an orientation peak is formed followed by a gradual loss of alignment until a steady state value is reached. Analysis of Figure 4.5b, reveals the development of another pair of non-orthotropic components in $\mathbb{A}^{\bar{\mathrm{dev}}4}$. For this flow, (d_8, d_9) appear throughout the flow history, with their magnitudes increasing as C_I rises.



(a) Time evolution of A_{ij} components at different levels of C_I

(b) Time evolution of d_i components at different levels of C_I

Figure 4.5: Temporal evolution of A_{ij} and d_i components under the homogeneous velocity field $\mathbf{v}_2$. Dash-dot line $\longrightarrow C_I = 0$, dashed line $\longrightarrow C_I = 1 \times 10^{-3}$, solid line $\longrightarrow C_I = 1 \times 10^{-2}$.

The contribution of $\mathbb{A} : \mathbf{D}$ to the orientation dynamics using flow $\mathbf{v}_2$ is given by Equation (4.1), where all components of $\mathbb{A}$ are included in the result. However, throughout the flow history, the components $A_{1123}, A_{1131}, A_{2223}, A_{2231}, A_{3323}, A_{3331}$ have a null value.

$$\mathbb{A} : \mathbf{D} = \begin{bmatrix} A_{1111}\epsilon + A_{1112}\gamma + A_{1122}\epsilon - 2A_{3311}\epsilon \\ A_{1122}\epsilon + A_{2212}\gamma + A_{2222}\epsilon - 2A_{2233}\epsilon \\ A_{2233}\epsilon + A_{3311}\epsilon + A_{3312}\gamma - 2A_{3333}\epsilon \\ \sqrt{2}\left(A_{1123}\epsilon + A_{2223}\epsilon + A_{2231}\gamma - 2A_{3323}\epsilon\right) \\ \sqrt{2}\left(A_{1123}\gamma + A_{1131}\epsilon + A_{2231}\epsilon - 2A_{3331}\epsilon\right) \\ \sqrt{2}\left(A_{1112}\epsilon + A_{1122}\gamma + A_{2212}\epsilon - 2A_{3312}\epsilon\right) \end{bmatrix} \tag{4.1}$$

Figure 4.6 shows the decomposition of the active components of the contraction for this flow into linear and orthotropic contributions, along with the its non-orthotropic components. The results indicate that, for this case, the non-orthotropic pair has only a marginal influence on the orientation dynamics, with the combined effect of the linear and orthotropic terms accounting the nearly the entire contribution.

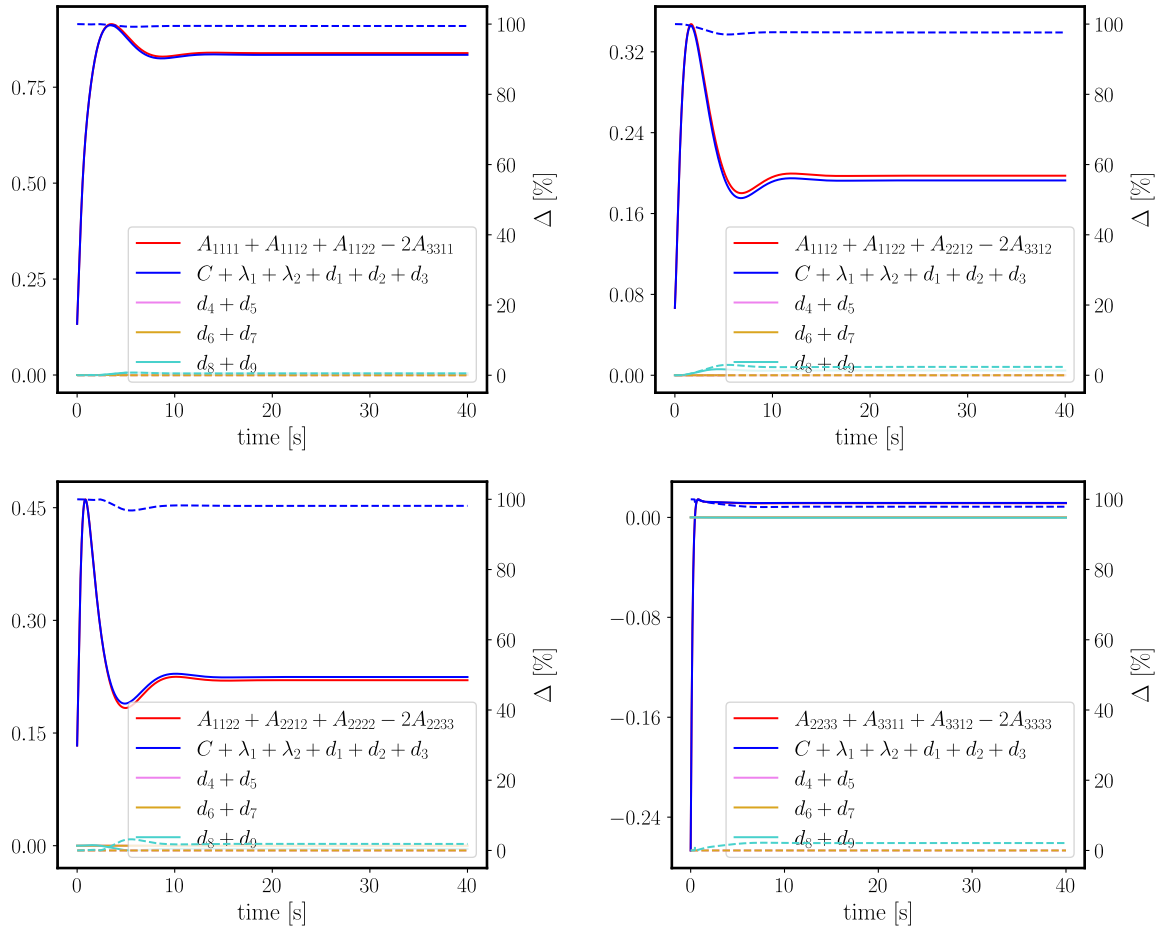


Figure 4.6: Split of active components of $\mathbb{A} : \mathbf{D}$ in Canonical space for flow $\mathbf{v}_2$ at $C_I = 1 \times 10^{-2}$. Dashed lines correspond to the relative contribution of each component.

Except for the initial isotropic state, the third principal direction of $\mathbf{A}$ remains aligned with that of $\mathbf{D}$ throughout the flow history. The evolution of the principal directions of $\mathbf{A}$ corresponds to a rotation in the $\mathbf{e}_1^{\mathrm{A}} - \mathbf{e}_2^{\mathrm{A}}$ plane, Figure 4.7.

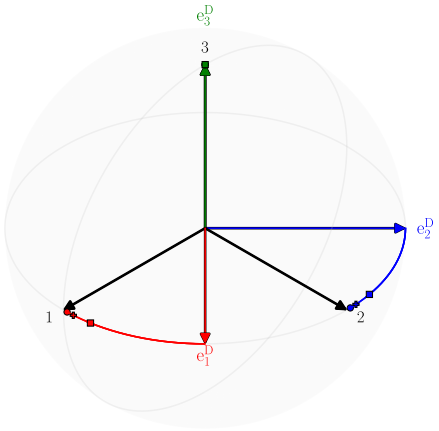


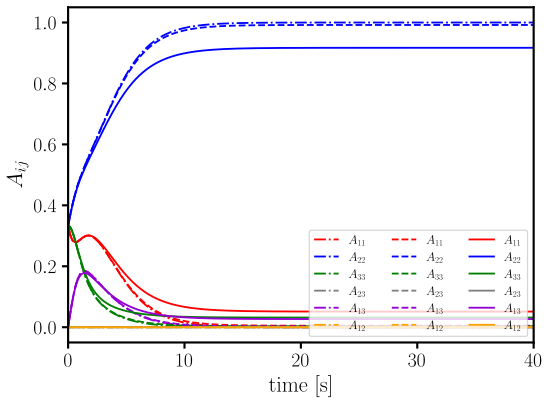
Figure 4.7: Trajectory of principal directions of $\mathbf{A}$ for flow $\mathbf{v}_2$. The principal directions $\mathbf{e}_1^A$, $\mathbf{e}_2^A$, and $\mathbf{e}_3^A$ are represented in red, blue, and green, respectively. The final position along each trajectory are marked with a circle, cross, and square for $C_I = 0$, $C_I = 1 \times 10^{-3}$, and $C_I = 1 \times 10^{-2}$, respectively.

4.3.3 Planar flow with shear

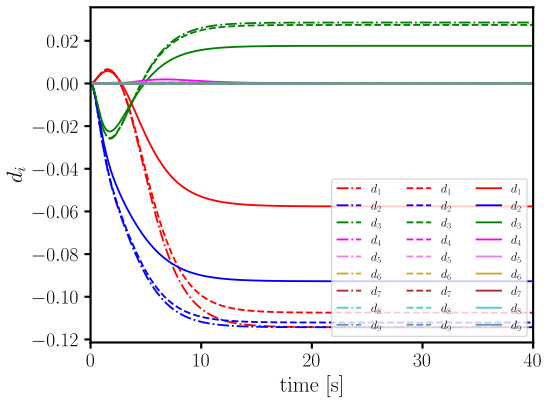
Planar flows constitute a subclass of extensional flows in which one direction experiences no deformation, causing a material element to be stretched along one direction while contracting along another. The introduction of shear into the velocity field can produce a profile that is representative of an isothermal, Newtonian flow in a center-gated disk [9]. In this section we analyze the flow-induced fiber orientation obtained under a homogeneous velocity field $\mathbf{v}_3 = \begin{bmatrix} -\dot{\epsilon}x_1 + \dot{\gamma}x_3, & \dot{\epsilon}x_2, & 0 \end{bmatrix}^{\mathrm{T}}$, with $\dot{\gamma} = \frac{5}{4}$ and $\dot{\epsilon} = \frac{1}{2}$. The planar component produces a contraction in the x_1 direction and equal stretch along x_2, while the shear component produces deformation in the $x_1 - x_3$ plane.

Figure 4.8a depicts the transient evolution of the A_{ij} components. Fiber orientation develops rapidly along the stretch direction, maintaining a consistent overall trend across all diffusion levels. The components A_{22} and A_{33} exhibit complementary dynamics with the former steadily increasing and the latter decreasing throughout the flow history. For dilute and intermediate diffusion levels, the A_{22} component approaches unity, indicating that planar flows are effective in producing a strong alignment in the stretch direction. In this range, the moderate level of diffusion is only marginally effective at limiting the degree of alignment, whereas a more pronounced reduction is observed for the highest diffusion level. For this flow, no orientation overshoot occurs, instead, A_{22} increases monotonically until reaching a steady-state value.

The evolution of the d_i components, depicted in Figure 4.8b, reveal the appearance of an additional pair of non-orthotropic components, d_4 and d_5, although their duration in the flow history is short. The orthotropic trio exhibit largely monotonic behavior, similar to the A_{ij} components. Similarly to the evolution of the second-order tensor, a moderate diffusion can only marginally alters the trajectories of the orthotropic trio, whereas higher diffusion levels cause a more pronounced deviation.



(a) Time evolution of A_{ij} components at different levels of C_I



(b) Time evolution of d_i components at different levels of C_I

Figure 4.8: Temporal evolution of A_{ij} and d_i components under the homogeneous velocity field $\mathbf{v}_3$. Dash-dot line $\longrightarrow C_I = 0$, dashed line $\longrightarrow C_I = 1 \times 10^{-3}$, solid line $\longrightarrow C_I = 1 \times 10^{-2}$.

For this case, and given the short contribution of the $(d_4 - d_5)$ pair, the contraction $\mathbb{A} : \mathbf{D}$ is dominated by the linear component and the orthotropic trio, and is therefore not examined in more detail.

Throughout the flow history, excluding the initial isotropic state, the first principal direction of $\mathbf{A}$ remains aligned with that of $\mathbf{D}$. The evolution of the principal directions of A_{ij} corresponds to a rotation in the $\mathbf{e}_2^{\mathrm{A}} - \mathbf{e}_3^{\mathrm{A}}$ plane, Figure 4.9.

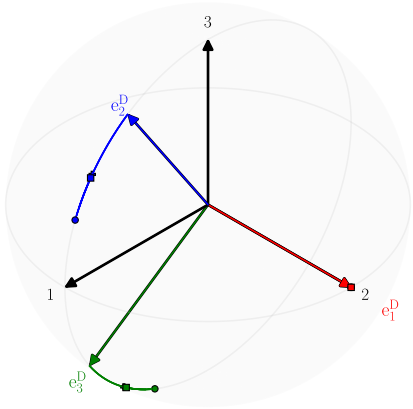


Figure 4.9: Trajectory of principal directions of $\mathbf{A}$ for flow $\mathbf{v}_3$. The principal directions $\mathbf{e}_1^{\mathrm{A}}$, $\mathbf{e}_2^{\mathrm{A}}$, and $\mathbf{e}_3^{\mathrm{A}}$ are represented in red, blue, and green, respectively. The final position along each trajectory are marked with a circle, cross, and square for $C_I = 0$, $C_I = 1 \times 10^{-3}$, and $C_I = 1 \times 10^{-2}$, respectively.

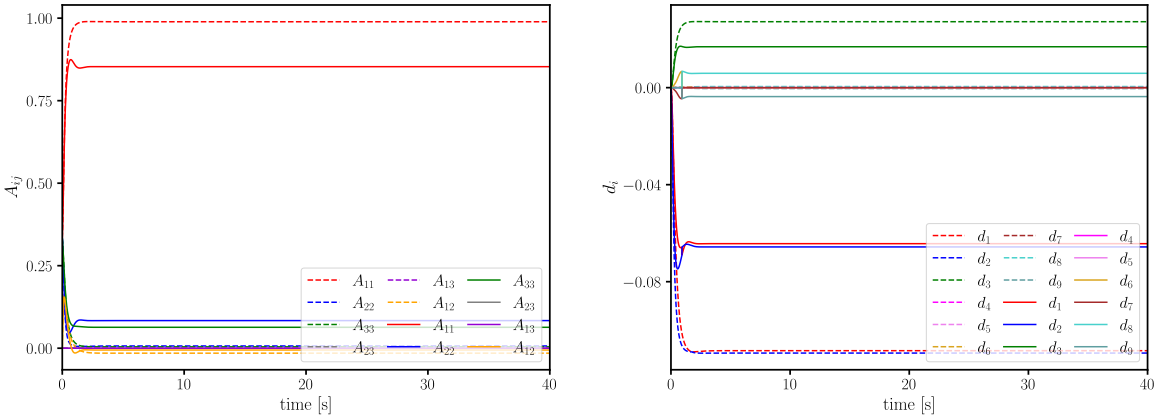
4.3.4 Uniaxial flow with shear

Uniaxial flows represent a specific category of extensional flows, characterized by the stretching of material along one direction while uniformly contracting it along the remaining two. In such flows,

fibers are expected to align in the stretching direction [11].

In this section, we examine two uniaxial flows combined with shear. The first defined by the velocity field $\mathbf{v}_4 = \left[2x_1 + 10x_2, \quad -x_2, \quad -x_3\right]^{\mathrm{T}}$, and the second by $\mathbf{v}_5 = \left[-x_1 + 10x_2, \quad -x_2, \quad 2x_3\right]^{\mathrm{T}}$.

The first velocity field stretches the material along the x_1 direction and shears it in the $x_1 - x_2$ plane. The corresponding evolution of A_{ij} and d_i components is shown in Figure 4.10. At a moderate level of diffusion, the A_{11} component rises rapidly towards a unitary value with a small exposure to the flow. Similarly to the previous analyzed flow, this moderate level of diffusion only marginally prevents the orientation from full alignment. At the highest diffusion level, a small orientation overshoot appears in the A_{11} trajectory, followed by a gradual decrease until a steady state is reached.



(a) Time evolution of A_{ij} components at different levels of C_I

(b) Time evolution of d_i components at different levels of C_I

Figure 4.10: Temporal evolution of A_{ij} and d_i components under homogeneous velocity field $\mathbf{v}_4$. Dashed line $\longrightarrow C_I = 1 \times 10^{-3}$, solid line $\longrightarrow C_I = 1 \times 10^{-2}$.

The evolution of the $\mathbb{A}^{\bar{\mathrm{dev}}4}$ components reveals several noteworthy features. At a diffusion level of $C_I = 1 \times 10^{-3}$, the orthotropic trio contributes almost entirely to $\mathbb{A}^{\bar{\mathrm{dev}}4}$. As C_I increases, two pairs of non-orthotropic components, specifically (d_6, d_7) and (d_8, d_9), become visible. Starting from the isotropic state, and consistent with the prior observations, the appearance of $d_6 - d_7$ pair occurs when $\mathbf{e}_2^{\mathrm{A}}$ is aligned with $\mathbf{e}_2^{\mathrm{D}}$. At $t \approx 0.93$ s a transition occurs in which this pair is replaced by $d_8 - d_9$, accompanied by a marked change in the d_1 and d_2 components. Data analysis shows that this transition coincides with the evolution of λ_3 surpassing that of λ_2, resulting in an interchange of the associated principal directions. This is illustrated in Figure 4.11, where the switchover point and associated transition in principal direction are highlighted. As depicted in Figure 4.1, the trajectory inside the UBT triangle for this flow appears to intersect the TU edge at the moment the switch in dynamics occurs.

The contribution of $\mathbb{A} : \mathbf{D}$ for the orientation dynamics is given in Equation (4.1). Although all components of $\mathbb{A}$ appear in the contraction, $A_{1123}, A_{1131}, A_{2223}, A_{2231}, A_{3323}$ and A_{3331} are null

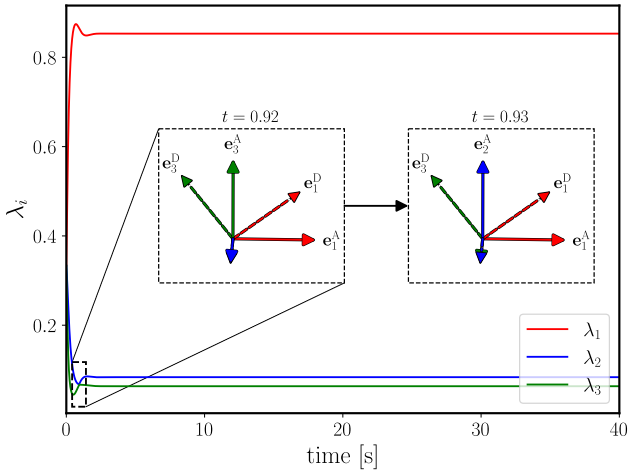


Figure 4.11: Eigenvalues evolution for under the homogeneous flow $\mathbf{v}_4$ with $C_I = 1 \times 10^{-2}$

throughout the flow history.

$$\mathbb{A} : \mathbf{D} = \begin{bmatrix} 2A_{1111}\epsilon + A_{1112}\gamma - A_{1122}\epsilon - A_{3311}\epsilon \\ 2A_{1122}\epsilon + A_{2212}\gamma - A_{2222}\epsilon - A_{2233}\epsilon \\ -A_{2233}\epsilon + 2A_{3311}\epsilon + A_{3312}\gamma - A_{3333}\epsilon \\ \sqrt{2}\left(2A_{1123}\epsilon - A_{2223}\epsilon + A_{2231}\gamma - A_{3323}\epsilon\right) \\ \sqrt{2}\left(A_{1123}\gamma + 2A_{1131}\epsilon - A_{2231}\epsilon - A_{3331}\epsilon\right) \\ \sqrt{2}\left(2A_{1112}\epsilon + A_{1122}\gamma - A_{2212}\epsilon - A_{3312}\epsilon\right) \end{bmatrix} \tag{4.1}$$

The contributions of $C, \lambda_{1,2}, d_{1\ldots9}$ using flow $\mathbf{v}_4$ to the contraction is depicted in Figure 4.12. Analysis of the Figure indicates that the non-orthotropic components have a non-negligible contribution to the contraction operation. This effect is less pronounced for the dynamics of A_{11} and A_{13}, but more pronounced for the remaining diagonal components.

The evolution of the eigenvectors of $\mathbf{A}$ are depicted in Figure 4.13. Prior to the switchover point, the eigenvectors evolution corresponds to a rotation in the the $\mathbf{e}_1^\mathbf{D}$–$\mathbf{e}_3^\mathbf{D}$ plane. Beyond this point, their evolution is minimal, with $\mathbf{e}_3^\mathbf{A}$ gradually aligning with $\mathbf{e}_2^\mathbf{D}$.

The velocity field $\mathbf{v}_5$ induces a stretch along the x_3 direction, while shearing in the $x_1 - x_2$ plane. The evolution of the A_{ij} and d_i components is illustrated in Figure 4.14. Similar to $\mathbf{v}_4$, the dynamics of orientation evolve rather rapidly. Starting from the isotropic state, the A_{11} component initially rises due to the influence of the shear flow, which aligns the fibers along the flow direction, in this case x_1. However, as the exposure to the flow increases, the effects of uniaxial extension become predominant, leading to a rapid increase of the A_{33} component. The only non-zero off-diagonal component is A_{12}, which exhibits a rapid initial rise followed by a gradual decay toward a small steady-state value.

Analysis of the evolution of the d_i components, depicted in Figure 4.14b, shows the appearance of two additional non-orthotropic pairs during the flow, (d_6, d_7) and (d_4, d_5). As in the case of $\mathbf{v}_4$

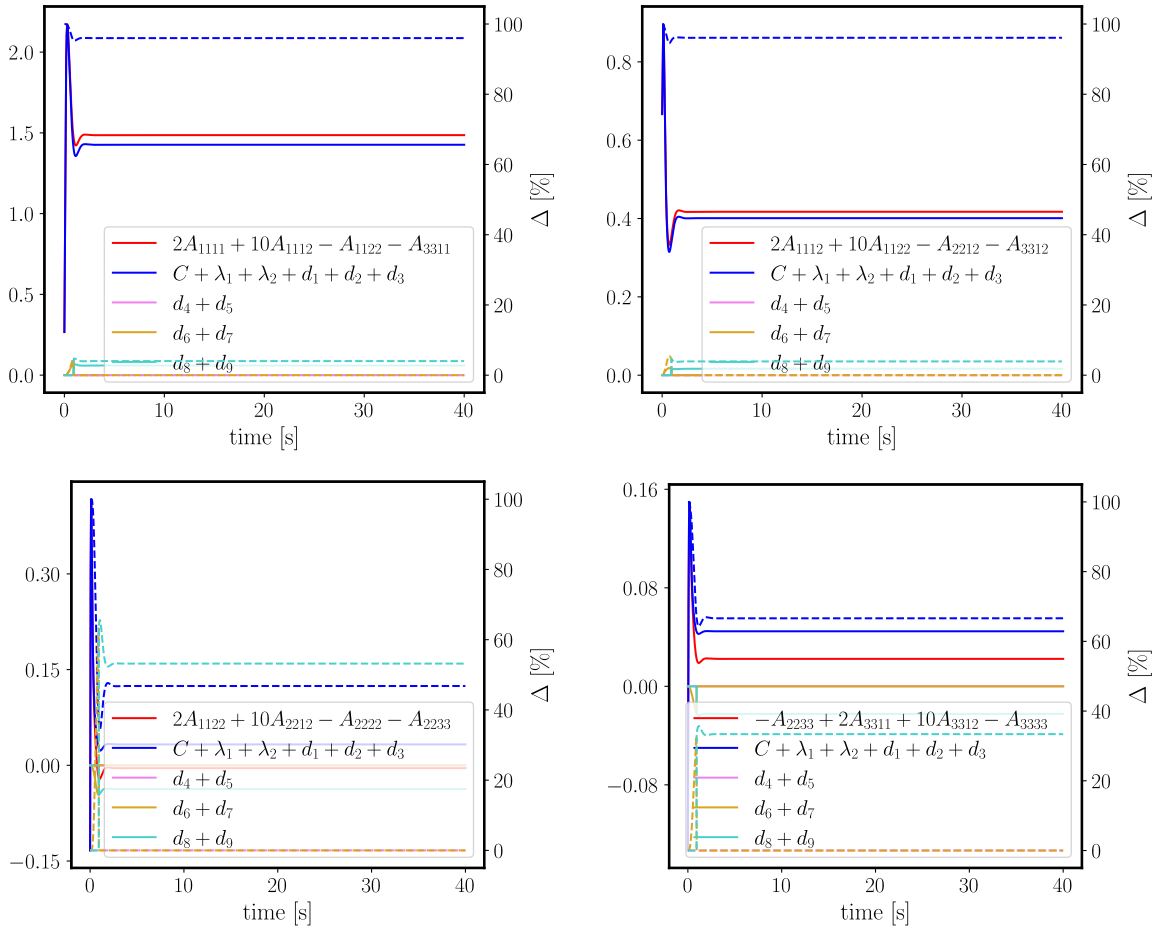


Figure 4.12: Split of active components of $\mathbb{A} : \mathbf{D}$ in Canonical space for flow $\mathbf{v}_4$ at $C_I = 1 \times 10^{-2}$

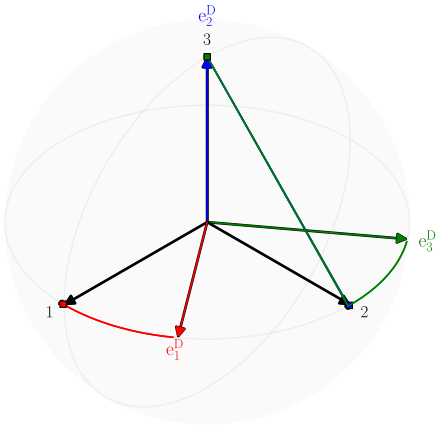
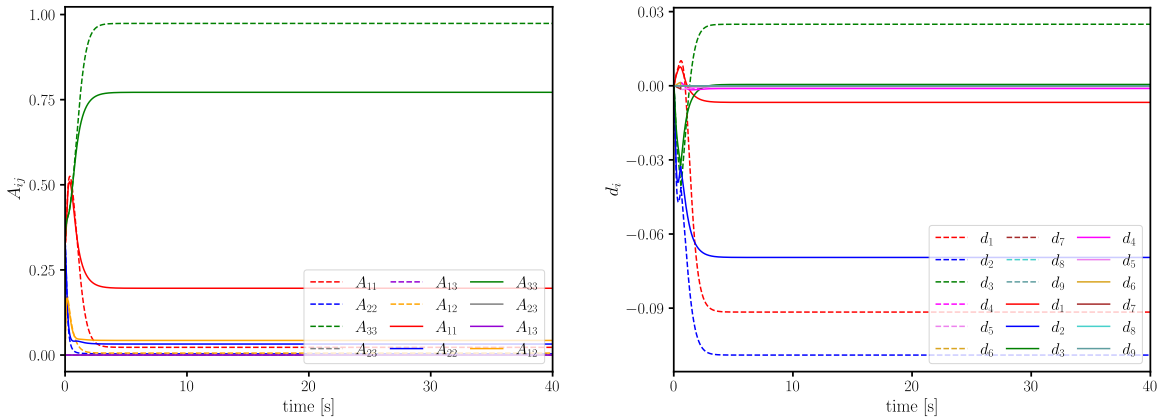


Figure 4.13: Trajectory of principal directions of $\mathbf{A}$ for flow $\mathbf{v}_4$. The principal directions $\mathbf{e}_1^A$, $\mathbf{e}_2^A$, and $\mathbf{e}_3^A$ are represented in red, blue, and green, respectively. The final position along each trajectory are marked with a cross, and square for $C_I = 1 \times 10^{-3}$ and $C_I = 1 \times 10^{-2}$, respectively.

a switchover point in the dynamics of the eigenvalues is observed in Figure 4.15. For this flow, the dynamics associated with λ_2 takeover those of λ_1, with $\mathbf{e}_1^A \parallel \mathbf{e}_2^D$. Furthermore, Figure 4.1, shows that the trajectory in the UBT triangle appears to intersect the TB edge at the moment this shift in



(a) Time evolution of A_{ij} components at different levels of C_I

(b) Time evolution of d_i components at different levels of C_I

Figure 4.14: Temporal evolution of A_{ij} and d_i components under homogeneous velocity field $\mathbf{v}_4$. Dashed line $\rightarrow C_I = 1 \times 10^{-3}$, solid line $\rightarrow C_I = 1 \times 10^{-2}$.

dynamics occurs.

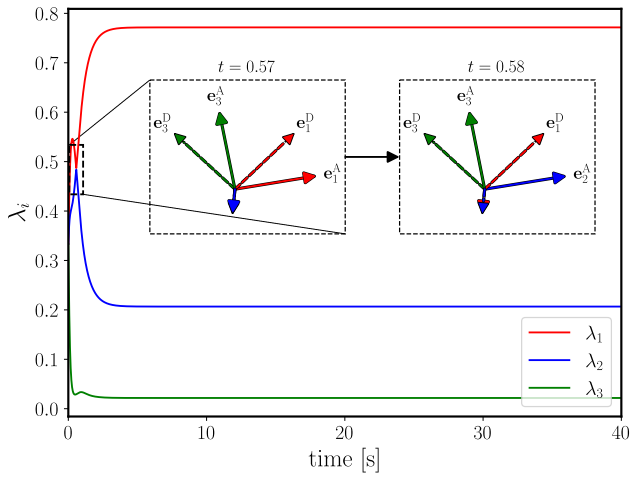


Figure 4.15: Eigenvalues evolution for under the homogeneous flow $\mathbf{v}_5$ with $C_I = 1 \times 10^{-2}$

Although the non-orthotropic components for flow $\mathbf{v}_5$ are not null, their effect on the contraction operation, $\mathbb{A} : \mathbf{D}$ is minimal. The dynamics of orientation for $\mathbf{v}_5$ are determined by the linear portion and the orthotropic trio. Therefore, their contribution to the flow induced orientation will not be examined any further.

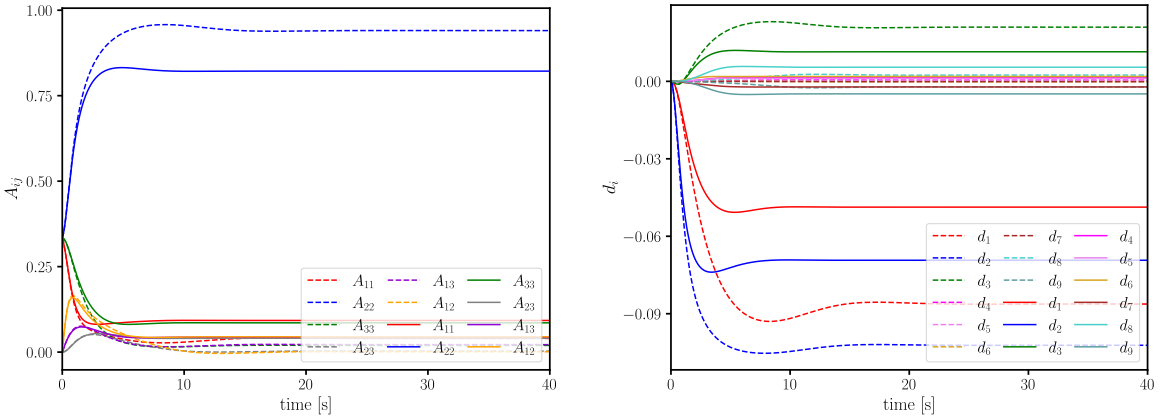
4.3.5 Other

As a final example, we analyze the effect of flow induced orientation generated by a planar flow combined with two shear flows in different planes, defined by $\mathbf{v}_6 = \left[-\frac{1}{4}x_1 + \frac{1}{2}x_3, \quad \frac{1}{4}x_2 + 2x_1, \quad 0\right]^{\mathrm{T}}$.

This velocity field produces an orientation state that is not aligned with any of the Cartesian coordinate axes.

The evolution of the A_{ij} and the fourth-order deviatoric tensor is depicted in Figure 4.16. For this velocity field, the dominant orientation develops along the stretch direction. At both medium and high diffusion levels, the A_{22} component rises quickly to form an orientation peak before slightly decaying to a steady-state value. The other diagonal components rapidly decrease to align with the increasing dynamics of A_{22}. For this flow, all non-diagonal components are active, indicating that the principal direction of orientation lies in a non-planar configuration.

Figure 4.16b shows the evolution of the deviatoric components of $\bar{\mathbb{A}}$. All non-orthotropic components develop, with d_8 and d_9 showing a more pronounced contribution. Analysis of the eigenvector evolution of $\mathbf{A}$ reveals no alignment with the principal directions of $\mathbf{D}$ throughout the flow history is observed.



(a) Time evolution of A_{ij} components at different levels of C_I

(b) Time evolution of d_i components at different levels of C_I

Figure 4.16: Temporal evolution of A_{ij} and d_i components under homogeneous velocity field $\mathbf{v}_6$. Dashed line $\rightarrow C_I = 1 \times 10^{-3}$, solid line $\rightarrow C_I = 1 \times 10^{-2}$.

The contraction contribution to the evolution dynamics is described by Equation (4.1), which involves 13 active components of $\mathbb{A}$. None of these maintain a null value throughout the flow history.

$$\mathbb{A} : \mathbf{D} = \begin{bmatrix} -\frac{A_{1111}}{4} + 2A_{1112} + \frac{A_{1122}}{4} + \frac{A_{1131}}{2} \\ -\frac{A_{1122}}{4} + 2A_{2212} + \frac{A_{2222}}{4} + \frac{A_{2231}}{2} \\ \frac{A_{2233}}{4} - \frac{A_{3311}}{4} + 2A_{3312} + \frac{A_{3331}}{2} \\ \frac{\sqrt{2}(-A_{1123}+A_{2223}+8A_{2231}+2A_{3312})}{4} \\ \frac{\sqrt{2}(8A_{1123}-A_{1131}+A_{2231}+2A_{3311})}{4} \\ \frac{\sqrt{2}(-A_{1112}+8A_{1122}+2A_{1123}+A_{2212})}{4} \end{bmatrix} \tag{4.1}$$

The contribution of each term in $\mathbb{A} : \mathbf{D}$ for the 4 components is illustrated in Figure 4.17. Analysis of the figure reveals several antagonistic effects among the non-orthotropic components. In

the dynamics of A_{22}, only the pair (d_8, d_9) contributes significantly, albeit modestly. Conversely, a more pronounced effect of $d_{4...9}$ can be observed in the remaining diagonal and off-diagonal components.

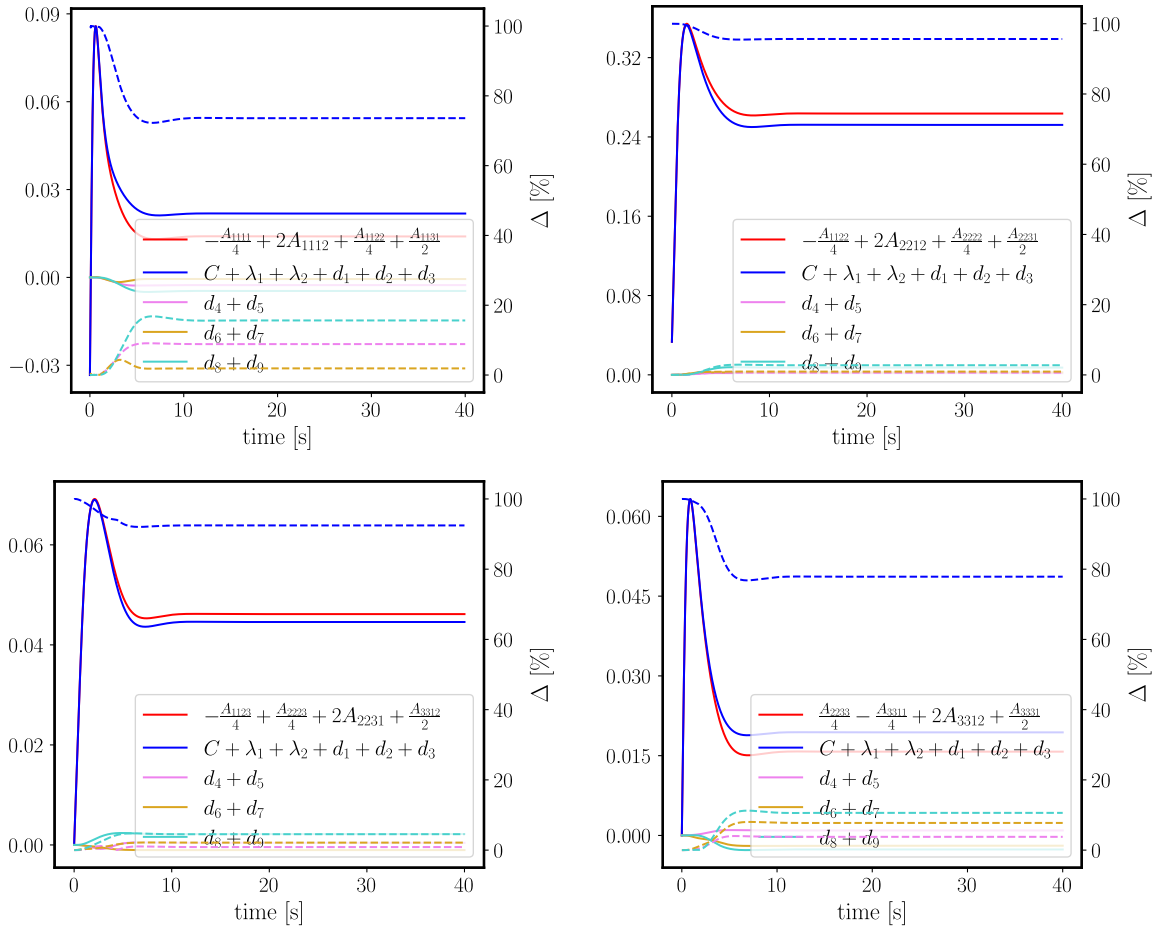


Figure 4.17: Split of active components of $\mathbb{A} : \mathbf{D}$ in Canonical space for flow $\mathbf{v}_6$ at $C_I = 1 \times 10^{-2}$. Dashed lines correspond to the relative contribution of each component.

The evolution of the eigenvectors for this flow is depicted in Figure 4.18. The trajectories of the principal direction are non-planar, reflecting the complex three-dimensional relation between stretching and rotation provided by $\mathbf{v}_6$. At the highest level of diffusion it can be observed a deviation in the path of $\mathbf{e}_1^{\mathbf{A}}$ and $\mathbf{e}_2^{\mathbf{A}}$ in regards to the moderate level.

This section demonstrates that flow-induced orientation in the presence of isotropic rotary diffusion can yield fourth-order tensors that deviate from orthotropic symmetry, even though orthotropy captures much of the orientation dynamics. The magnitude of this deviation increases with the level of diffusion in the system. To more accurately capture the real behavior of $\mathbb{A}$, it is essential to enrich the closure relations with additional objective information. Consistent with the findings of Tucker [13], we suggest that a more effective closure relations can be developed by incorporating additional information derived directly from the flow kinematics. Our observations under a range of flow conditions are summarized in Table 4.1. In systems where diffusion and shear act concurrently, non-orthotropic components are likely to develop. This information is crucial for the development of

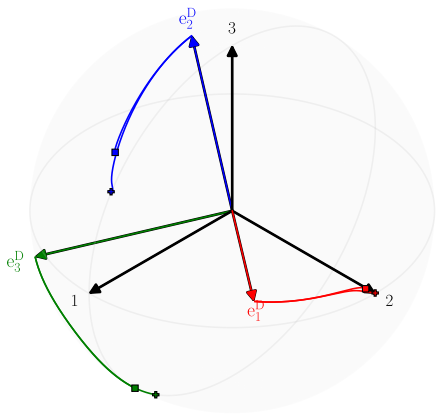


Figure 4.18: Trajectory of principal directions of $\mathbf{A}$ for flow $\mathbf{v}_6$. The principal directions $\mathbf{e}_1^{\mathrm{A}}$, $\mathbf{e}_2^{\mathrm{A}}$, and $\mathbf{e}_3^{\mathrm{A}}$ are represented in red, blue, and green, respectively. The final position along each trajectory are marked with a cross and square for $C_I = 1 \times 10^{-3}$, and $C_I = 1 \times 10^{-2}$, respectively.

better closures, which should integrate other objective flow kinematics information.

Table 4.1: Resume of observed transitions in non-orthotropic components

$\mathbf{e}_1^A \parallel \mathbf{e}_1^D$	(d_4, d_5)
$\mathbf{e}_2^A \parallel \mathbf{e}_2^D$	(d_6, d_7)
$\mathbf{e}_3^A \parallel \mathbf{e}_3^D$	(d_8, d_9)
$\mathbf{e}_3^A \parallel \mathbf{e}_2^D$	(d_8, d_9)
$\mathbf{e}_1^A \parallel \mathbf{e}_2^D$	(d_4, d_5)
$\mathbf{e}_i^A \nparallel \mathbf{e}_j^D$	$(d_4, d_5, d_6, d_7, d_8, d_9)$

Under the assumption that the orthotropic symmetry can be relaxed, we provide empirical limits for an admissible monoclinic symmetry in Appendix B.

4.4 Evaluation of orientation dynamics

To evaluate the effect of constraining $\mathbb{A}$ to be orthotropic when expressed in the principal axes of orientation, we conduct numerical experiments using the ORE closure [11] and a polynomial model fitted using only the data in simple-shear with $C_I = 1 \times 10^{-2}$. Another classical literature model, the IBOF [9], could have been used, however, it yields a similar level of accuracy to the ORE model, as such, the latter was selected.

Recognizing that the fourth-order tensor can be decomposed into linear and deviatoric components, and in line with other fitted closures [9, 10, 16], we employed a polynomial model to describe the deviatoric components of $\mathbb{A}$, as a function of the independent eigenvalues, thus $d_{1\ldots9} \approx f(\lambda_1, \lambda_2)$. The fitting was performed using the Scikit-learn library v.1.5.1 [25] to obtain a

third-order polynomial without the constant term. To enforce the initial isotropic initial condition, a built-in weighting option was used to give more importance to the isotropic point during fitting.

Figure 4.19 shows the results of the fitting procedure. Visually, the curves are fairly well matched, with minor deviations observed in the d_6, d_7 trajectories within the $[7.5 - 15]$ s interval. Numerically, the root mean squared (RMS) error is approximately 7.85×10^{-5}, suggesting that this approach is likely to yield accurate results.

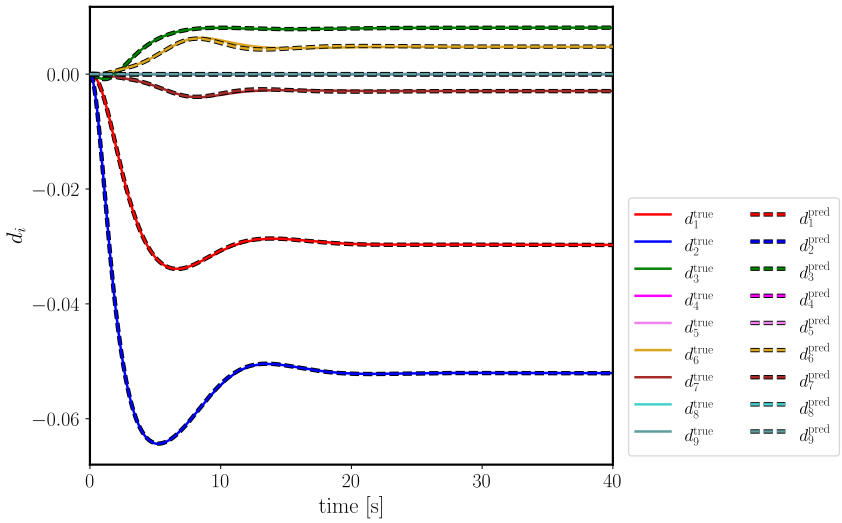


Figure 4.19: Prediction of d_i coefficients from a third-order polynomial model fitted with data in simple-shear with $C_I = 1 \times 10^{-2}$

Equation (2.7) was solved numerically using the `solve_IVP` routine of Scipy [26] with a Runge-Kutta 5(4) scheme [27] employing an absolute and relative tolerance of 1×10^{-10}.

A comparison among the state-of-the-art closure, the polynomial model incorporating only the orthotropic trio, and the polynomial model using all d_i components is depicted in Figure 4.20. Analysis of the figure shows that the ORE model accurately follows the true dynamics of the system during during the early stages of the flow history. However, once the orientation overshoot in the true dynamics starts to decrease, the ORE fails to capture the correct behavior. Instead of reproducing the orientation peak, the orientation in the A_{11} component increasing monotonically until reaching a steady-state, which is modestly over-predicted.

The polynomial model restricted to the orthotropic trio, can also accurately follow the A_{ij} trajectory during the initial flow history. Similarly to the ORE model, upon the development of the orientation peak in the true dynamics, an overshoot in the A_{11} component is observed with a formation on an orientation peak latter in the flow history. Although this model successfully predicts the presence of an overshoot, it misrepresents the true dynamics of A_{ij} with a broader transition period. At steady-state, this model outperforms the ORE counterpart, but still over-predicts the A_{11} value.

The results from the polynomial model that predicts all d_i components, exhibit dynamics that

closely match the true dynamics, effectively capturing all relevant features of the flow-induced orientation.

Given that the ORE was fitted for several types of flows, it will be excluded from further discussion. Although this analysis might seem trivial, it highlights the significance of the non-orthotropic components in identifying a closure that achieves greater accuracy.

When fitting a polynomial model, it is assumed that the eigenvalues computed from the integration of Equation (2.7) will be similar to those used to derive the polynomial model. This assumption holds when the flow kinematics and the $\mathbb{A}$ constraints are well aligned. Flows such as simple shear, where non-orthotropic components have a small but non-negligible influence, the orthotropic assumption becomes too restrictive and the computed fourth-order tensor fails to guide the solution toward the correct dynamics, resulting in systematic errors during integration.

The orthotropic assumption proposed by [10] accurately reflects the intended properties of the stiffness tensor and is exact in the dilute regime. However, as diffusion in the system increases, the accuracy of this assumption becomes more nuanced. When aiming to obtain an orthotropic fourth-order tensor for a constitutive equation, while preserving the correct trajectory of the A_{ij} components, a polynomial fitting may not be the most effective approach, as illustrated in Figure 4.20. An alternative and potentially more effective method, would be a neuralODE [28] to determine an equivalent orthotropic tensor that can maintain the correct dynamics of A_{ij}. Our experience suggests that while this approach may perform well for a single flow type, it fails to generalize effectively to other flow configurations if the orthotropic constrain is not relaxed.

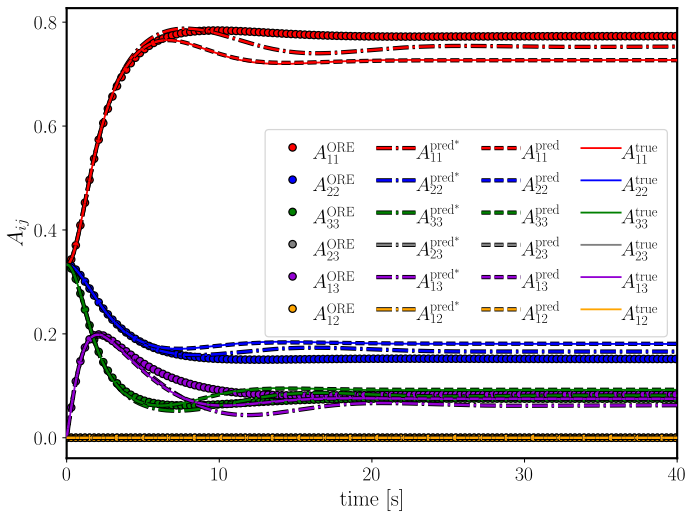


Figure 4.20: Prediction of the A_{ij} dynamics in simple-shear flow with $C_I = 1 \times 10^{-2}$ using different closure models. A_{ij}^{true} represents the true dynamics of the system, A_{ij}^{pred} model includes all d_i components, $A_{ij}^{\mathrm{pred}^*}$ includes only the orthotropic trio and A_{ij}^{ORE} corresponds to the ORE closure

.

4.5 Conclusion

In this work, selected flows scenarios were used to generate flow-induced probability distribution functions through the numerical solution of the Folgar-Tucker model at different diffusion levels. From these, the second and fourth-order orientation tensor where computed and the latter decomposed using the harmonic decomposition framework to extract their deviatoric components.

Traditionally, closures used to approximate the fourth-order tensor rely on directional information solely provided by the second moment tensor under the assumption of orthotropy. The findings of this work show that for specific shear flows, where diffusion is present, the resulting fourth-order tensor expressed in the principal frame of orientation, exhibits non-orthotropic components. The magnitude of these is influenced by both the level of diffusion and the type of flow. At high diffusion levels, their contribution to the orientation dynamics can be significant, suggesting that the orthotropic restriction may need to be relaxed to accurately capture the evolution of the second-order tensor.

Analysis of the obtained data reveals that the non-orthotropic components developing during the flow history can be inferred from the alignment of the principal directions of the rate-of-deformation tensor with those of the second-order orientation tensor. This information is expected to facilitate the development of new three-dimensional closure relations that more accurately capture the orientation dynamics in these simplified flows scenarios.

Appendices

Appendix A Data generation with Stochastic approach

Jeffery's equation of motion is the only mechanistic component of the orientation dynamics. As previously mentioned, modeling fiber-fiber interactions involves adding a phenomenological diffusive flux to the continuity equation of the ODF. This can also be accomplished by including a random walk into Equation (2.1), [21, 29], allowing it to be recast as a stochastic differential equation in the following form:

$$\mathrm{d}\mathbf{p} = \dot{\mathbf{p}}\,\mathrm{d}t + \sigma\,\mathrm{d}\mathbf{W}, \tag{A.1}$$

where, σ is a scalar diffusion coefficient and $\mathrm{d}\mathbf{W}$ is the differential of a Wiener process.

Different approaches for modeling a random walk on the surface of the unit sphere were discussed by Novikov and co-workers [30]. One such approach, as proposed by Manhart [21], uses a random walk produced by a number generator with a uniform distribution. In the context of an explicit

Euler integrator, after computing the solution at $\mathbf{p}^{t+1}$, a perturbation is introduced via the random walk and the final orientation vector is obtained by normalizing the perturbed orientation vector with its magnitude:

$$\mathbf{p}^{t+1} = \frac{\mathbf{p}_J^{t+1} + \Delta\mathbf{p}}{\left\|\mathbf{p}_J^{t+1} + \Delta\mathbf{p}\right\|}, \tag{A.2}$$

$$\Delta\mathbf{p} = \chi\mathbf{X}, \tag{A.3}$$

$$\mathbf{X} = U\left(-\frac{1}{2}, \frac{1}{2}\right) \text{ and} \tag{A.4}$$

$$\chi = \sqrt{24 D_r \Delta t}, \tag{A.5}$$

where $\mathbf{p}_J^{t+1}$ is Jeffery orientation vector at $\mathbf{p}^{t+1}$, $D_r = C_I\dot{\gamma}$ is the scalar diffusion coefficient and $\mathbf{X}$ is a vector with components uniformly distributed on the interval $\left[-\frac{1}{2}, \frac{1}{2}\right]$.

For a discrete number of points over the conformational space and sampling-rate, the second-order orientation tensor can be computed as follows:

$$\mathbf{A} = \frac{1}{N_{\text{pts}}} \sum_{l=1}^{N_{\text{pts}}} \frac{1}{N_{\text{samples}}} \sum_{k=1}^{N_{\text{samples}}} p_i^k p_j^k, \tag{A.6}$$

where N_{pts} is the number of points sampling the unit sphere and N_{samples} is the number of samples at each point.

A comparison between data generated using the ODF solution and stochastic method is depicted in Figure 21, showing the temporal evolution of the components A_{11} and A_{13} under homogeneous simple-shear flow with $C_I = 1 \times 10^{-2}$. To ensure statistical reliability, $N_{\text{pts}} = 50\,000$ points were sampled over the unit sphere following the procedure in [24]. At each point, Equation (A.1) was solved for $N_{\text{samples}} = 3\,000$ using a time-step of 2×10^{-3} s. The figure shows an overlap of the data from both approaches, with a maximum absolute difference between the second-order data of $\approx 1.3 \times 10^{-4}$, indicating that both methods are suitable for synthetic data generation.

Appendix B Parametric bounds

In line with the strategy of [31], we constrain the search space of admissible fourth-order tensors assuming a monoclinic symmetry by enriching the orthotropic structure with additional components. Admissible empirical bounds where determined using the software *Mathematica* 14® [32] for the pairs (d_4, d_5), (d_6, d_7) and (d_8, d_9) based on Sylvester's criterion to ensure that the 6x6 matrix representing the 4^{th}-order tensor stays positive-definite. Taking the pair (d_6, d_7) as an example, with analogous derivation for the remaining non-orthotropic pairs, and using the flows reported in [31] with

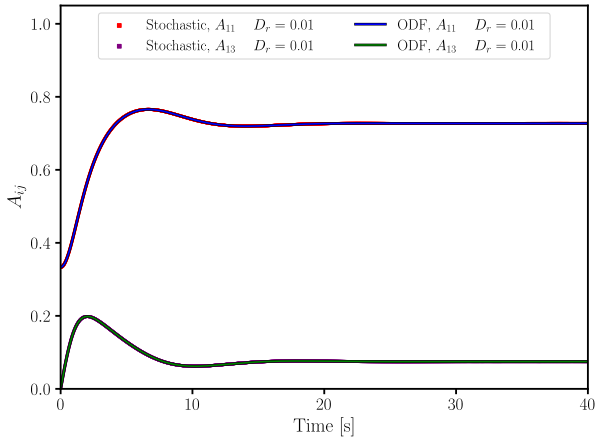


Figure 21: Comparison between ODF and stochastic approach in simple shear $\mathbf{v} = \begin{bmatrix} x_3, & 0, & 0 \end{bmatrix}^{\mathrm{T}}$ with $C_I = 1 \times 10^{-2}$

a $C_I = 1 \times 10^{-2}$, the admissible values were determined by considering $\bar{\mathbb{A}}$ as:

$$\tilde{\bar{\mathbb{A}}} = \begin{bmatrix} a & b & c & 0 & \sqrt{2}d_6 & 0 \\ b & d & e & 0 & -\sqrt{2}\,(d_6 + d_7) & 0 \\ c & e & f & 0 & \sqrt{2}d_7 & 0 \\ 0 & 0 & 0 & 2e & 0 & -2d_6 - 2d_7 \\ \sqrt{2}d_6 & -\sqrt{2}\,(d_6 + d_7) & \sqrt{2}d_7 & 0 & 2c & 0 \\ 0 & 0 & 0 & -2d_6 - 2d_7 & 0 & 2b \end{bmatrix} \tag{B.1}$$

where a, b, c, d, e, f are the numerical values of the orthotropic entries.

Our interpretation of the solution of Sylvester's criterion indicates that the admissible region is given by the intersect region refined by the 2 elliptical solutions and the 2 linear solutions of $\det\left(\tilde{\bar{\mathbb{A}}}\right) = 0$ when solved with respect to d_7, Equation (B.2).

$$s_1 = \frac{\alpha d_6 - \sqrt{\beta + d_6^2 \gamma}}{\delta} \tag{B.2a}$$

$$s_2 = \frac{\alpha d_6 + \sqrt{\beta + d_6^2 \gamma}}{\delta} \tag{B.2b}$$

$$s_3 = -d_6 - \sqrt{be} \tag{B.2c}$$

$$s_4 = -d_6 + \sqrt{be} \tag{B.2d}$$

where $\alpha, \beta, \gamma, \delta$ are coefficients given as function of the orthotropic entries:

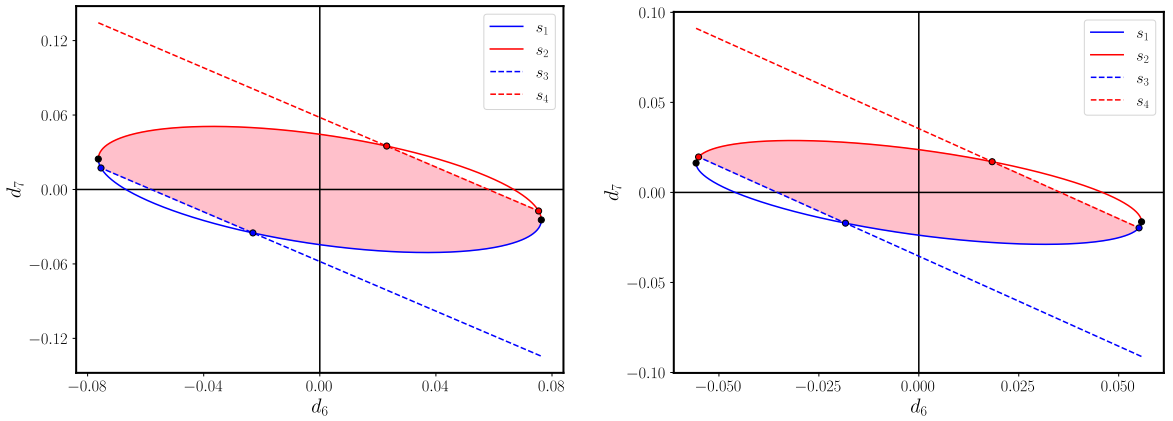
$$\alpha = -ae - af + bc - be - bf + c^2 + cd + ce \tag{B.3}$$

$$\begin{aligned}\beta = {} & a^2cd^2f - a^2cde^2 + 2a^2cdef + a^2cdf^2 - 2a^2ce^3 - a^2ce^2f - 2ab^2cdf \\ & + ab^2ce^2 - 2ab^2cef - ab^2cf^2 + 2abc^2de - 2abc^2df + 6abc^2e^2 + 2abc^2ef \\ & - ac^3d^2 - 2ac^3de - 2ac^3df + ac^3e^2 + b^4cf - 2b^3c^2e + 2b^3c^2f \\ & + b^2c^3d - 4b^2c^3e + b^2c^3f + 2bc^4d - 2bc^4e + c^5d\end{aligned} \tag{B.4}$$

$$\begin{aligned}\gamma = {} & -a^2df + a^2e^2 + ab^2f - 2abce - 2abdf + 2abe^2 + ac^2d - 2acdf \\ & + 2ace^2 - ad^2f + ade^2 - 2adef - adf^2 + 2ae^3 + ae^2f + 2b^3f \\ & - 4b^2ce + 2b^2cf + b^2df + 2b^2ef + b^2f^2 + 2bc^2d - 4bc^2e \\ & - 2bcde - 4bce^2 - 2bcef + 2c^3d + c^2d^2 + 2c^2de + c^2df\end{aligned} \tag{B.5}$$

$$\delta = ad + 2ae + af - b^2 - 2bc - c^2 \tag{B.6}$$

Examples of admissibility regions are given in Figure 22.



(a) $\lambda_1 = 0.5213$, $\lambda_2 = 0.30934$, $d_1 = -0.00359$, $d_2 = -0.01725$, $d_3 = -0.00085$

(b) $\lambda_1 = 0.73574$, $\lambda_2 = 0.18084$, $d_1 = -0.02968$, $d_2 = -0.05208$, $d_3 = 0.00806$

Figure 22: Admissibility region, in pink, for the pair (d_6, d_7) for simple-shear flow with $C_I = 1 \times 10^{-2}$ at 1 s and 25 s, respectively. Values are rounded to the 5^{th} decimal case

Appendix C Availability of data and material

The data that support the findings of this study are openly available at `https://github.com/Computational-Rheology/FOM_Non_ortho_contrib`.

References

[1] Suresh G. Advani and III Tucker, Charles L. The use of tensors to describe and predict fiber orientation in short fiber composites. *Journal of Rheology*, 31(8):751–784, 11 1987.

[2] Jin Wang, John F. O'Gara, and Charles L. Tucker. An objective model for slow orientation kinetics in concentrated fiber suspensions: Theory and rheological evidence. *Journal of Rheology*, 52(5):1179–1200, 09 2008.

[3] Jay H. Phelps and Charles L. Tucker. An anisotropic rotary diffusion model for fiber orientation in short- and long-fiber thermoplastics. *Journal of Non-Newtonian Fluid Mechanics*, 156(3):165–176, 2009.

[4] Huan-Chang Tseng, Rong-Yeu Chang, and Chia-Hsiang Hsu. An objective tensor to predict anisotropic fiber orientation in concentrated suspensions. *Journal of Rheology*, 60(2):215–224, 03 2016.

[5] Alexander Bakharev, Huagang Yu, Shishir Ray, Russell Speight, and Jin Wang. Using new anisotropic rotational diffusion model to improve prediction of short fibers in thermoplastic injection molding. In *Annual Technical Conference - ANTEC, Conference Proceedings*, volume 2018-May, 2018.

[6] Huan-Chang Tseng, Rong-Yeu Chang, and Chia-Hsiang Hsu. The use of principal spatial tensor to predict anisotropic fiber orientation in concentrated fiber suspensions. *Journal of Rheology*, 62(1):313–320, 01 2018.

[7] Susanne Katrin Kugler, Argha Protim Dey, Sandra Saad, Camilo Cruz, Armin Kech, and Tim Osswald. A flow-dependent fiber orientation model. *Journal of Composites Science*, 4(3), 2020.

[8] Suresh G. Advani and III Tucker, Charles L. Closure approximations for three□dimensional structure tensors. *Journal of Rheology*, 34(3):367–386, 04 1990.

[9] Du Hwan Chung and Tai Hun Kwon. Invariant-based optimal fitting closure approximation for the numerical prediction of flow-induced fiber orientation. *Journal of Rheology*, 46(1):169–194, 01 2002.

[10] Jr. Cintra, Joaquim S. and III Tucker, Charles L. Orthotropic closure approximations for flow□induced fiber orientation. *Journal of Rheology*, 39(6):1095–1122, 11 1995.

[11] Charles L. Tucker III. *Fundamentals of Fiber Orientation: Description, Measurement and Prediction*. Hanser Publishers, 2022.

[12] Najam ul Qadir and David A. Jack. Modeling fibre orientation in short fibre suspensions using the neural network-based orthotropic closure. *Composites Part A: Applied Science and Manufacturing*, 40(10):1524–1533, 2009.

[13] Charles L. Tucker. Planar fiber orientation: Jeffery, non-orthotropic closures, and reconstructing distribution functions. *Journal of Non-Newtonian Fluid Mechanics*, 310:104939, 2022.

[14] G.G. Lipscomb, M.M. Denn, D.U. Hur, and D.V. Boger. The flow of fiber suspensions in complex geometries. *Journal of Non-Newtonian Fluid Mechanics*, 26(3):297–325, 1988.

[15] Stephen Montgomery-Smith, Wei He, David A. Jack, and Douglas E. Smith. Exact tensor closures for the three-dimensional jeffery's equation. *Journal of Fluid Mechanics*, 680:321–335, 2011.

[16] Brent Eugene VerWeyst. *Numerical Predictions of Flow-Induced Fiber Orientation in Three-Dimensional Geometries*. PhD thesis, University of Illinois at Urbana-Champaign, 1998.

[17] George L. Hand. A theory of anisotropic fluids. *Journal of Fluid Mechanics*, 13(1):33–46, 1962.

[18] Randy Scott Bay. *Fiber orientation in injection-molded composites: A comparison of theory and experiment*. Phd thesis, University of Illinois at Urbana-Champaign, Example City, CA, 1991. Permalink: `http://hdl.handle.net/2142/22217`.

[19] Fransisco Folgar and Charles L. Tucker III. Orientation behavior of fibers in concentrated suspensions. *Journal of Reinforced Plastics and Composites*, 3(2):98–119, 1984.

[20] Julian Karl Bauer and Thomas Böhlke. Variety of fiber orientation tensors. *Mathematics and Mechanics of Solids*, 27(7):1185–1211, 2022.

[21] Michael Manhart. Rheology of suspensions of rigid-rod like particles in turbulent channel flow. *Journal of Non-Newtonian Fluid Mechanics*, 112(2):269–293, 2003.

[22] Charles L. Tucker. Fiber Orientation Tools, 2022. `https://github.com/charlestucker3/Fiber-Orientation-Tools`.

[23] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[24] Edward B Saff and Amo BJ Kuijlaars. Distributing many points on a sphere. *The mathematical intelligencer*, 19:5–11, 1997.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[26] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas,

Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[27] J.R. Dormand and P.J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.

[28] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations, 2019.

[29] Amin Moosaie and Michael Manhart. A direct numerical simulation method for flow of brownian fiber suspensions in complex geometries. *Journal of Dispersion Science and Technology*, 34(3):427–440, 2013.

[30] A. Novikov, D. Kuzmin, and O. Ahmadi. Random walk methods for monte carlo simulations of brownian diffusion on a sphere. *Applied Mathematics and Computation*, 364:124670, 2020.

[31] Bruno Ramoa, Chady Ghnatios, João Miguel Nóbrega, and Francisco Chinesta. Physics-informed data-driven closure relation for dilute short fiber suspensions. *Adv. Model. Simul. Eng. Sci.*, 12(1), March 2025.

[32] Wolfram Research, Inc. Mathematica, Version 14.0. Champaign, IL, 2024.

Chapter 5 Rank Reduction Autoencoders applied to fiber orientation dynamics

Abstract:

In the realm of composite materials, traditional methods for modeling fiber orientation represent the state-of-orientation (SoO) using macro-descriptors such as orientation tensors. While these descriptors can effectively capture microstructural variability, they often fail to accurately describe the actual fiber orientation, even in simple shear-driven flows that are relevant to industrial practices.

In this work, we propose the usage of a Rank-Reduction Autoencoder (RRAE) coupled with a multilayer perceptron (MLP) to infer compact, physics-aware orientation descriptors capable of representing the steady-state SoO in planar flows. The RRAE compresses the orientation distribution function (ODF) into a low-dimensional latent space and reconstructs it via truncated singular-value decomposition, retaining only the most significant orientation modes. The learning process is guided by a multi-component loss function that balances the reconstruction of the ODF, the suspension stress, and, for consistency with conventional approaches, the second-order orientation tensor. Concurrently, the suspension stress is learned through a data-driven enrichment added to a known Newtonian baseline.

The model was trained on synthetic data that was generated under non-dilute conditions using the Folgar-Tucker model over a broad parametric domain, and benchmarked against the Hybrid and Natural closure relations. The latter is exact in the absence of diffusion. Using only three leading modes, the proposed framework achieved superior average accuracy for both stress and orientation predictions, demonstrating that a low-rank, data-driven representation can effectively replace conventional closure models. To further enhance usability, a surrogate MLP mapping process parameters directly to the latent space coefficients was trained, enabling fast, ODF-free inference for steady-state orientation and stress prediction.

Keywords: Fiber orientation modeling; Rheology and flow driven; Rank Reduction; Autoencoders

---

This chapter was adapted from B. Ramoa, J. Mounayer, J.M. Nóbrega, and F. Chinesta, Rank Reduction Autoencoders applied to fiber orientation dynamics, submitted to the Journal of Engineering Applications of Artificial Intelligence (2025)

5.1 Introduction

Orientation modeling is essential for predicting the anisotropic behavior of fiber-reinforced materials, particularly polymer composites, as it directly correlates with the material's thermo-rheological properties and ultimately in its mechanical performance.

This topic has been actively researched over the past decades [1–7], driven by the role of CAD/CAE software in modern engineering. Phenomenological models have steadily increased in complexity to better capture experimental observations.

Due to computational requirements, the main state-of-the-art models used in CAE applications rely on macro-descriptors of orientation, known as orientation tensors [8–10]. These descriptors are derived from a richer mesoscale descriptor, known as the orientation distribution function [6, 11], and provide a concise representation of the material's microscopic structure [12].

Models using these descriptors typically focus on the second-order orientation tensor as the primary quantity of interest [1] to balance computational efficiency and accuracy for engineering applications.

However, this macro-descriptor is known to be non-unique as different orientation distribution functions can yield the same macro-descriptor [13] and presents a closure problem [1] in its evolution equation, arising from the need to approximate the next even-order tensor based on the lower-order descriptor. Several formulations addressing these issue have been proposed in the literature [12–15] based on mechanistic, fitted, or data-driven relations.

The contribution of fiber orientation to the bulk stress of a suspension is of paramount importance due to the strong coupling between orientation and flow during the manufacturing of thermoplastic composites. The suspension introduces an anisotropic contribution to the bulk stress of the solvent/matrix, which depends on the state of orientation [1]. This is commonly described in the literature through the constitutive equation proposed by Dinh and Armstrong [16].

Recent advances in machine learning (ML) and computational power have opened new avenues for addressing the limitations of traditional modeling approaches. Combining known constitutive models with ML technologies can lead to data-driven models that better represent the non-linear behavior of materials [17].

Training such models can require a vast amount of experimental or synthetic data. Given the high-dimensionallity of the rich meso-scale orientation descriptor, a technique that extracts the most important features from the data while reducing dimensionallity, would be beneficial for orientation modeling.

Autoencoders are a type of ML technology that maps high-dimensional input to a learned low-dimensional space. This lower-dimensional space, referred to as the "latent space", retains the most important features of the input data. This is traditionally achieved through two main

components: an encoder that learns the most significant features of the data and represents them in a lower-dimensional latent space, and a decoder that reconstructs the original input from the compressed representation of the data [18]. This technology has been applied in various fields, including data compression [19], medicine [20], polymer science [21], among others [22].

Traditional autoencoders have two limitations: i) the degree of compression is directly related to the size of the latent space, which must be chosen before hand, leading to trial-and-error experiments that can be challenging depending on the nature of the data, and ii) a lack of regularization in the latent representation can create "holes" in their lower-dimensional representation [18].

To address both limitations, a new class of autoencoders, known as Rank Reduction Autoencoders (RRAE), has been recently developed [18]. The main innovation of this architecture lies in the integration of Singular Value Decomposition (SVD) to the latent space while training. This approach regularizes the latent representation and constrains the bottleneck to the SVD coefficients of the latent space. This allows the ranking of the most significant dimensions, which improves the efficiency and interpretability of the latent space. Moreover, because of to the batching nature utilized in the training procedure, this technique introduces a regularization effect that stabilizes the learning process.

In this work, our goals are twofold: i) to obtain a compact latent representation of the ODF whose singular values are influenced not only by the reconstruction of the ODF but also by the suspension stress, and ii) leveraging the ranking of the bottleneck such that a linear combination of the singular values captures the second-order orientation tensor due to its widespread use in engineering computations.

We demonstrate the applicability and effectiveness of the developed framework in planar orientation, given the acceptable accuracy of planar orientation in sheet molding compound [23], at steady-state.

For simplicity, we assume Newtonian behavior of the matrix ($\boldsymbol{\tau}^{\mathrm{phy}}$), and infer an enrichment to $\boldsymbol{\tau}^{\mathrm{phy}}$ by modeling the epistemic uncertainty/ignorance, $\boldsymbol{\tau}^{\mathrm{enrich}}$, using the experimental data, $\boldsymbol{\tau}^{\mathrm{exp}}$. Mathematically this is expressed as:

$$\boldsymbol{\tau}^{\mathrm{enrich}} = \boldsymbol{\tau}^{\mathrm{exp}} - \boldsymbol{\tau}^{\mathrm{phy}}. \tag{5.1}$$

Given the unknown volume of data required to infer $\boldsymbol{\tau}^{\mathrm{enrich}}$, synthetic data is employed in this work, assuming the orientation dynamics follow the isotropic rotary diffusion model proposed by Folgar and Tucker [24], to obtain the experimental data ($\boldsymbol{\tau}^{\mathrm{exp}}$). Leveraging the flexibility of the synthetic datasets, we explore a broad parametric space to train the model.

The remainder of the paper is organized as follows: Section 5.2 details the methodology used to generate the synthetic dataset. Section 5.3 describes the architecture of the neural network model along with its training procedure. In Section 5.4, we present and analyze the obtained results. In

Section 5.5 we broaden the functionality of the RRAE by creating a parameter-to-coefficient surrogate to bypass the input necessary to train the RRAE. Finally, Section 5.6 summarizes the main conclusions of this work and outlines future research directions.

5.2 Data generation

The synthetic dataset was generated by solving the Fokker-Plank equation in two-dimensional space [25] up to steady-state, Equation (2.4a), using the isotropic rotary diffusion model, Equation (2.8).

The governing equation was discretized using a power-law scheme [26] on a grid with 512 uniformly spaced nodes over $[-\pi/2, \pi/2[$ using a grid spacing of $\Delta\phi = \pi/512$ and employing periodic boundary conditions [25, 26]. An implicit Euler scheme was used to integrate the governing equation, starting from a uniform distribution ($\psi_0 = 1/2\pi$) with a constant velocity gradient, a unitary value for ξ, and a grid Courant number of 0.1. During the calculation procedure, a steady-state was considered to be achieved if the difference between $\left|\psi^{t+1} - \psi^{t}\right| < 1 \times 10^{-7}$ over 100 consecutive time-steps. For the calculation, the freely available integration routine coded in MATLAB programming language [27], was ported to Julia programming language [28].

Given the symmetry conditions of the ODF and the grid uniform spacing ($\Delta\phi$), the following normalization was employed $\psi^{\mathrm{norm}} = 2\psi\Delta\phi$, ensuring unitary integration of ψ^{norm} over the solution space. For simplicity, for the remainder of the manuscript we use ψ to denote ψ^{norm}, unless otherwise stated.

To explore a large parametric space in planar orientation, the data was generated by varying two parameters: i) a velocity gradient, and ii) the scalar diffusion coefficient. However, this framework must learn equivariance, thus a third parameter was used, iii) a rotation matrix, to help the network learn rotation-equivariant behavior and improve robustness.

The employed velocity gradient presents the following form [23]:

$$L = \begin{bmatrix} 0 & 1 \\ \alpha & 0 \end{bmatrix}, \tag{5.1}$$

where α controls the type of flow, with simple-shear attained for a null value, and planar extension and rigid-body rotation obtained for 1 and -1, respectively [23].

The rotation matrix is given by [29]:

$$Q(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}, \tag{5.2}$$

Table 5.1: Parameters used to generate ODF at steady-state

α	θ	C_I
$[-0.8, -0.1[$ in steps of 0.05 $[-0.1, 0.1[$ in steps of 0.0025 $[0.1, 1[$ in steps of 0.01	$\left[1 \times 10^{-3}, 2 \times 10^{-2}\right]$ in steps of 1×10^{-3}	$\left[0, \frac{19\pi}{20}\right]$ in steps of $\frac{\pi}{20}$

where θ denotes the rotation angle. The model must learn to recognize that different velocity gradients may produce similar ODFs, differing only by a shift along the ϕ axis. The final velocity gradient is computed by applying the rotation matrix to the velocity gradient, $\mathbf{L}_\theta = \mathbf{Q}(\theta) \cdot \mathbf{L} \cdot \mathbf{Q}(\theta)^{\mathrm{T}}$.

The range of values used for each of the three parameters is reported in Table 5.1. In total, the sampling produced a dataset with $74\,000$ orientation distribution curves at steady-state.

Assuming a fiber suspension in an incompressible Newtonian fluid, the extra-stress tensor, $\boldsymbol{\tau}$, can be computed through the constitutive equation of Dinh and Armstrong [16, 25], as:

$$\boldsymbol{\tau} = 2\eta \left[\mathbf{D} + N_p \mathbb{A} : \mathbf{D} + N_s \left(\mathbf{D} \cdot \mathbf{A}_2 + \mathbf{A}_2 \cdot \mathbf{D}\right)\right], \tag{5.3}$$

where η is the isotropic viscosity, and N_p and N_s are dimensionless parameters, known as the particle and shear number, respectively. These describe the anisotropic contributions from the suspended particles and do not depend on the state of orientation but rather on the fiber aspect-ratio and volume fraction [30]. For typical fiber aspect ratios used in composite materials, the particle number is significantly larger than the shear number; thus, the contributions from N_s are often neglected in the computation of the extra-stress tensor [25, 30].

Theoretical predictions for the particle number can be obtained in semi-concentrated suspensions based on the work of Dinh and Armstrong [16, 25], where N_p is obtained from:

$$N_p = \frac{\phi_f \left(l/d_f\right)^2}{3 \ln\left(2h_f/d_f\right)}, \tag{5.4}$$

with h_f representing the distance between a fiber and its nearest neighbors [25, 30], given by:

$$\frac{2h_f}{d_f} = \begin{cases} \sqrt{\pi/\phi_f} & \text{for aligned fibers} \\ \left(\pi d_f\right)/\left(2\phi_f l\right) & \text{for random fibers} \end{cases}, \tag{5.5}$$

where ϕ_f and l/d_f are the fiber volume fraction and aspect-ratio, respectively.

To compute N_p for the generated dataset, a volume fraction was estimated based on the model proposed by Phan-Thien and co-workers [31], where the isotropic rotary diffusivity is related to the

product of the fiber aspect-ratio (a_r) and volume fraction as follows:

$$C_I = 0.03 \left[1 - \exp\left(-0.244 a_r \phi_f\right)\right]. \tag{5.6}$$

Based on Equation (5.6), assuming unitary η, with selected values of C_I and $a_r = 30$, the corresponding value fraction was determined. This value was subsequently used to determine N_p and finally, $\boldsymbol{\tau}$ assuming an h_f value corresponding to aligned fibers. Notably, although a finite aspect-ratio was used to calculate ϕ_f, the ODFs were generated assuming a slender fiber, with $\xi = 1$. The option for a finite aspect-ratio was a modeling simplification, introduced to enable the estimation of the particle number using a known relationship from the literature.

To incorporate the stress into the RRAE framework, $\boldsymbol{\tau}$ was assumed to be $\boldsymbol{\tau}^{\text{exp}}$ and decomposed into $\boldsymbol{\tau}^{\text{phy}}$ and $\boldsymbol{\tau}^{\text{enrich}}$. The unknown component to be determined, is assumed to depend on the rate-of-deformation tensor $\mathbf{D}$, the particle number N_p, and the compressed representation of the ODF, through the latent space coefficients. Mathematically these relations are expressed as follows:

$$\boldsymbol{\tau}^{\text{exp}} = 2 \left[\underbrace{\eta \mathbf{D}}_{\boldsymbol{\tau}^{\text{phy}}} + \boldsymbol{\tau}^{\text{enrich}} \right], \tag{5.7}$$

where $\boldsymbol{\tau}^{\text{enrich}}$ is the ignorance of the model, that should be approximated with the proposed approach.

5.3 Training

To enable the stress to contribute to the inference of modes in the latent space, the overall model architecture combines a Reduced-Rank Autoencoder with a multi-layer perceptron (MLP) , forming the integrated RRAE-MLP network, illustrated in Figure 5.1. The architecture and training procedures were implemented using Python's programming language within the JAX framework [32].

Given the specific properties to enforce in the model, the loss function consisted of three terms, as shown in Equation (5.1), i) a reconstruction loss for the ODF, ii) a second-order orientation tensor loss, and iii) a stress tensor loss.

$$\boldsymbol{\mathcal{L}} = \begin{bmatrix} e_{\psi} & e_{\mathbf{A}_2} & e_{\boldsymbol{\tau}} \end{bmatrix} \tag{5.1}$$

The network takes as input ψ along with the auxiliary variables $\mathbf{D}$ and N_p. During the first step, the ODF is compressed into a lower-dimensional latent space, where SVD is applied to yield a basis and coefficients, which are truncated to k_{max} modes. Through experimentation, we found that $k_{max} = 3$

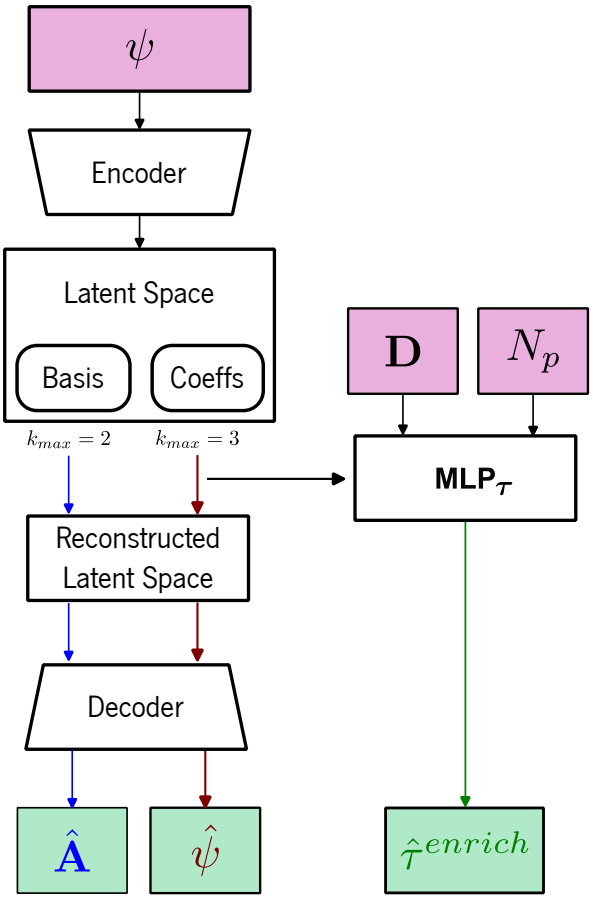


Figure 5.1: RRAE with MLP network. k_{max} defines the number of modes retained from SVD decomposition.

modes were sufficient to effectively reconstruct the planar ODF and the suspension stress. Currently, the RRAE technology allows the user to specify or adaptively determine the number of modes that best represent the input data [18].

After compressing the input into the latent space and performing the SVD, a reconstruction of the latent space with k_{max} modes is performed and fed into the decoder. The reconstruction loss is then calculated using the relative difference ratio between the input ODF and the model prediction, denoted by the superscript "$\wedge$", through a L^2-Norm, following Equation (5.2).

$$e_\psi = \frac{\left\|\psi - \hat{\psi}\right\|_2}{\|\psi\|_2} \tag{5.2}$$

To force the representation of the second-order tensor as a combination of modes in the latent space, we initially constrained the leading singular value to represent $\mathbf{A}_2$. However, the results obtained showed that a single mode was insufficient to adequately capture this quantity. Therefore, in subsequent tests we employed the first two modes of the latent space to capture $\mathbf{A_2}$. This quantity was computed by predicting an ODF with $k_{max} = 2$, followed by an integration over the conformational space, as shown in Equation (2.6). Similar to the reconstruction loss, we added a relative difference norm to the overall loss function. However, instead of the L^2-norm, the second

invariant of a tensor $\mathbf{T}$ was used, since it is commonly employed in previous works [23] as the error metric, following Equation (5.3).

$$
\begin{aligned}
e_{\mathbf{T}} &= \frac{f_{II}\left(\mathbf{T}-\hat{\mathbf{T}}\right)}{f_{II}\left(\mathbf{T}\right)}, \\
f_{II}\left(\mathbf{T}\right) &= \sqrt{2\mathbf{T}:\mathbf{T}}.
\end{aligned}
\tag{5.3}
$$

Concurrently, the obtained coefficients with k_{max}, together with $\mathbf{D}$ and N_p, were provided as input to the MLP. The network predicts the unknown component of the stress in the enriched constitutive equation. The associated stress‐loss term of the overall loss is defined following Equation (5.3).

Finally, the overall model loss was computed as a weighted average, with the weighting coefficients based on the relative contribution of each component to the total loss, as follows:

$$
\begin{aligned}
\mathcal{L} &= \sum w_i \mathcal{L}_i, \\
w_i &= \frac{\mathcal{L}_i}{\sum \mathcal{L}_i},
\end{aligned}
\tag{5.4}
$$

where w_i is the weighting coefficient for the i^{th} loss term, and $\mathcal{L}_i$ represents each term of the loss function.

A detailed description of each element of the network (encoder, decoder and MLP) is given in Table 5.2. All elements employed a feed-forward neural network with the rectified linear unit (ReLU) as the activation function for all layers, except for the output layer. In the encoder and MLP predicting the stress, the output layer used an identity mapping. For the decoder, since the ODF must be non-negative for all values of ψ and due to the normalization requirement, the softmax activation function was selected.

Table 5.2: Network structure

	Encoder	Decoder	MLP
width	30	30	24
depth	1	2	3
latent space size	30	-	-
k_{max}	3	-	-
Activation function	ReLU	ReLU	ReLU
Final activation function	Identity	softmax	Identity

Training of the RRAE-MLP model was carried out in two stages: i) standard training and ii) fine-tuning of the basis. The interested readers are referred to [18] for a more in-depth description of each stage. Unless otherwise stated, the learning process was performed with the default values of AdaBelief Optimizer [33] from Optax [34].

Standard training was performed in two stages, each lasting 1000 epochs, with a learning-rates of 1×10^{-3} and 1×10^{-4}, respectively. Afterwards, fine-tuning of the basis was carried out also in two stages, for 750 epochs, with learning-rates of 1×10^{-4} and 1×10^{-5}, respectively. The training data was divided into batches of size 20, with the loss function calculated as the average over the batch.

To infer the model's predictive capability, in an interpolation regime 300 points were removed from the training set and designated as the test set.

5.4 Discussion

5.4.1 Latent space

The obtained coefficient-space for the selected training parametric space is depicted in Figure 5.2. An exploded view was given to the condition $\theta = 0\ [rad]$, resulting in the omission of the corresponding values of c_3 from the plot, as they are not meaningful in this context. Considering the highlighted condition, the color of each line in the plot corresponds to a distinct value of α as a function of C_I. The size of the markers in the overlaid scatter plot corresponds to the magnitude of C_I, with larger markers representing greater values of the diffusion coefficient.

The structure observed in the figure highlights the ability of the RRAE to extract relevant low-dimensional features from the ODFs. In particular, the separation of trajectories associated with different flow types suggests that the latent representation encodes flow kinematics in a systematic manner. Meanwhile, the spread of points with varying diffusion coefficient C_I reveals that the model is sensitive to the balance between randomizing and flow-driven mechanisms of fiber orientation without any apparent folds or crossings in the coefficient space. Higher values of C_I drives the ODF towards a more isotropy state, thus points gather into a compact region with short inter-condition distance. As the diffusion value decreases, the flow-driven contribution becomes more relevant, anisotropic increases, and the trajectories expand along with the inter-condition distance. The regularity observed in the coefficient space is a crucial characteristic for interpolation as it enables the decoder to reconstruct ODFs and derived quantities with high fidelity, even when the data does not explicitly cover every intermediate configuration.

5.4.2 Comparison with known closures

Table 5.3 presents the relative errors for the dataset concerning both $\mathbf{A}_2$ and $\boldsymbol{\tau}$, comparing the RRAE-MLP model with the Natural and Hybrid closure models. Additionally, since the RRAE-MLP model also predicts the ODF, the corresponding values are also reported in the table. For the sake of fairness, the different ODF's are compared through a relative Wasserstein distance, following Equation (5.1).

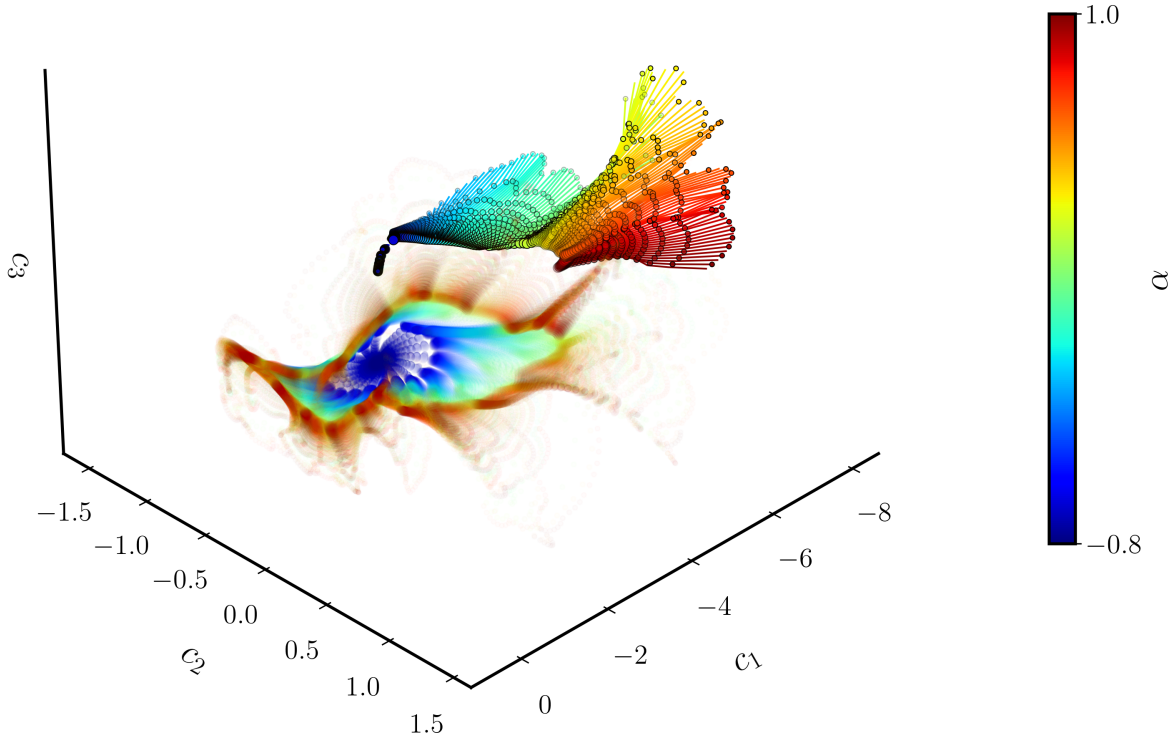


Figure 5.2: Coefficients of the latent space. Highlight for samples with $\theta = 0\,[rad]$.

$$l_1(u, v) = \frac{\int_{-\infty}^{+\infty} |U - V|}{\int_{-\infty}^{+\infty} |U|}, \tag{5.1}$$

where U and V represent the cumulative distribution functions of the ground truth and predicted ODFs, and $u = u(x)$, $v = v(x)$ denote the probabilities of each distribution at position x [35].

The errors computed for the closure models were obtained considering $\theta = 0\,[rad]$, as applying a rotation to $\mathbf{L}$ will produce a rotated state of orientation, $\mathbf{A}_{2_\theta} = \mathbf{Q}(\theta) \cdot \mathbf{A}_2 \cdot \mathbf{Q}(\theta)^{\mathrm{T}}$ [29]. Consequently, the computed relative errors are expected to remain unchanged with respect to θ.

Table 5.3: Errors in dataset for different models

	RRAE-MLP train [%]	RRAE-MLP test [%]	Natural [%]	Hybrid [%]
$\mathbf{A}_2$	min/max: [0.00 - 39.05] $\mu = 2.11$ $\sigma = 2.18$	min/max: [0.04 - 18.34] $\mu = 2.16$ $\sigma = 2.01$	min/max: [0.01 - 7.11] $\mu = 2.61$ $\sigma = 2.11$	min/max: [0.01 - 14.66] $\mu = 3.41$ $\sigma = 3.95$
$\boldsymbol{\tau}$	min/max: [0.04 - 19.57] $\mu = 2.55$ $\sigma = 1.64$	min/max: [0.32 - 11.27] $\mu = 2.65$ $\sigma = 1.71$	min/max: [0.01 - 22.85] $\mu = 5.22$ $\sigma = 6.41$	min/max: [0.05 - 46.80] $\mu = 12.19$ $\sigma = 13.72$
ψ	min/max: [0.01 - 39.77] $\mu = 1.07$ $\sigma = 1.77$	min/max: [0.03 - 9.13] $\mu = 1.01$ $\sigma = 1.24$	–	–

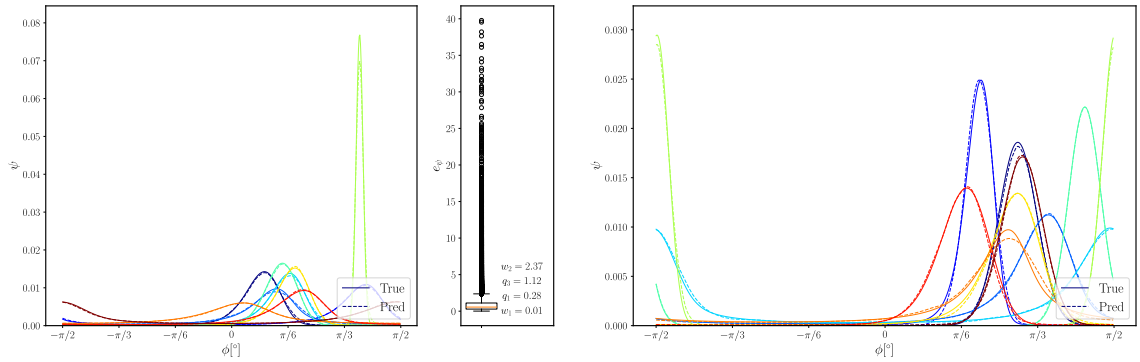
Analysis of Table 5.3 indicates that the RRAE-MLP model predicts the data fairly well within both the training and testing sets. The average relative errors are relatively small with less than 3.7% on

all predicted quantities. Considering the obtained results, the performance of the RRAE-MLP closure model clearly exceeds that of its conventional counterparts, Natural and Hybrid.

5.4.3 Prediction of ODF

To investigate the error distribution for $\hat{\psi}$, Figure 5.3 presents the ten predictions that are closest in value to the mean of the training set, accompanied by a box plot summarizing the relative error distribution, alongside with the ten nearest predictions to the upper whisker threshold.

The analysis of the box plot reveals that the first (Q_1) and third (Q_3) quartiles have values of approximately 0.28% and 1.12%, respectively. The whiskers extend to $\pm 1.5\,(Q_3 - Q_1)$, with any remaining data points classified as outliers in the box plot analysis. This indicates that a significant portion of the dataset exhibits a relatively small error. This is visually represented through the curves in both Figure 5.3a and 5.3b, where a strong visual agreement with the true data is evident.



(a) Top 10 ODFs closest to μ_ψ.

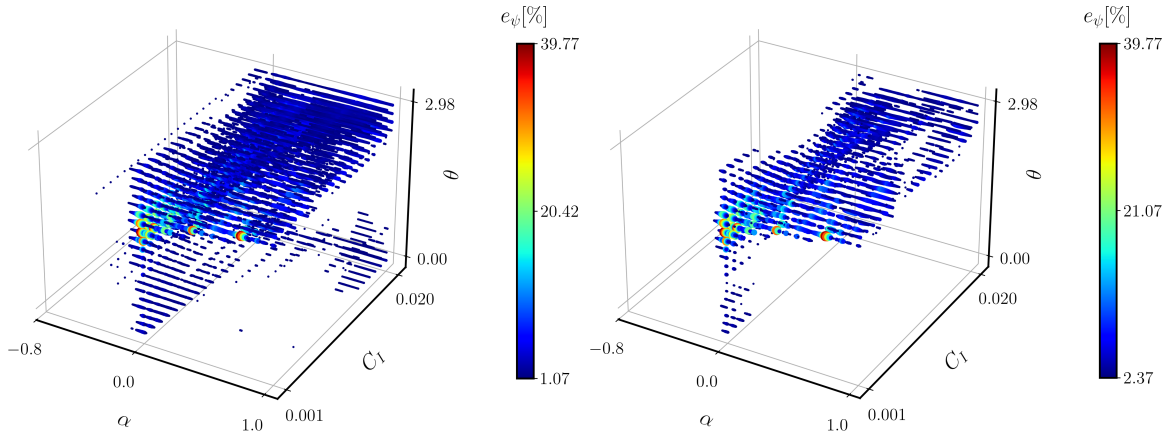
(b) Top 10 ODF closest to w_2.

Figure 5.3: Representative curves of ODF. True data is represented with a full line, predicted data with a dashed line.

To further analyze the error distribution of the developed model, Figure 5.4 shows the relative errors of the ODF over the parametric space, defined by α, C_I, and θ. Figure 5.4a indicates that most parameter combinations yield errors close μ_ψ with more significant issues identified at lower values of C_I, varying with θ and α. To isolate these problematic areas, Figure 5.4b depicts conditions exhibiting errors exceeding the top whisker of the box plot. At this threshold it is evident that only a limited number of conditions have substantial deviation from the true data. This relative error distribution highlights the non-invariant nature of the network, which is generally capable of capturing the rotational behavior across the parametric space, albeit with certain limitations in a few localized regions.

5.4.4 Prediction of Stress

The stress predicted by the RRAE-MLP model in the training set exhibits an average relative error (μ_τ) of 2.5%. This indicates that the MLP has effectively learned an overall accurate mapping between



(a) Values above μ

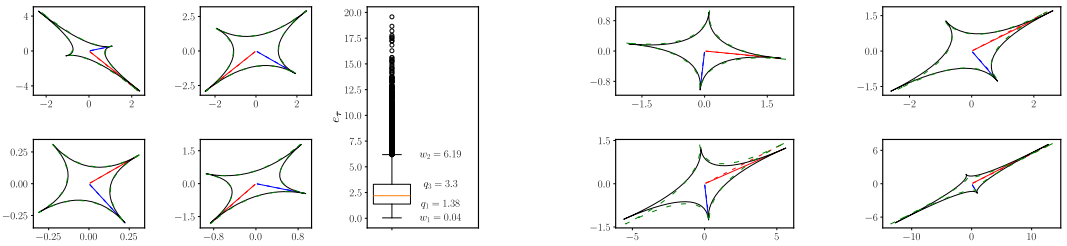
(b) Values above w_2

Figure 5.4: Error distribution of ψ over the parametric space.

the coefficients of the latent space, along with $\mathbf{D}$ and N_p, to predict the unknown stress component of the model.

Consistent with the methodology employed to predict the ODF, Figure 5.5 depicts four conditions whose values are closest to the average relative error, accompanied by a box plot analysis and the four conditions with relative error nearest to the top whisker.

To visualize the predicted symmetric tensors a superellipse (SE) representation [36] was adopted. In this, each tensor $\mathbf{T}$ is decomposed into its eigenvalues and eigenvectors. The eigenvalues define the length of the semi-axis, while the eigenvectors define the SE principal axis directions. When both eigenvalues are of the same sign, the SE takes on a convex form, colored orange or blue for positive and negative principal values, respectively. Conversely, if the eigenvalues are of opposite signs, the SE assumes a concave shape and is rendered in green.



(a) Representative values near μ_τ and box plot analysis.

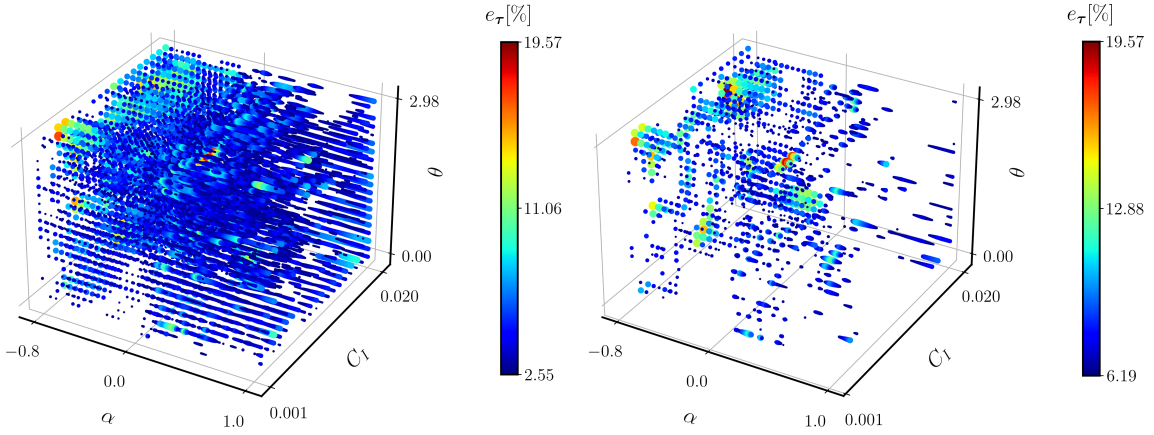
(b) Representative values near w_2

Figure 5.5: Prediction of $\boldsymbol{\tau}$

In the box plot presented in Figure 5.5a, the third quartile corresponds to a relative error of 3.3%, and the upper whisker extends to 6.2%, with higher values classified as outliers. The average error representations in Figure 5.5a indicates that the MLP model effectively reproduces the SE representation of the true stress tensor. However, at the upper end of the relative error spectrum, more significant discrepancies in the predicted principal values and directions are observed, although

the tensor representation remains reasonably accurate.

To further investigate problematic regions, Figure 5.6a illustrates the distribution of relative error across the training parametric space. For clarity, data points with errors below the average have been omitted from the visualization. In line with the box plot analysis, the figure demonstrates that the developed model yields accurate predictions across the majority of the parametric space. Notably, only 2600 points exhibit a relative error exceeding w_2, as highlighted in Figure 5.6b. Among these, the largest errors are concentrated at discrete locations $\alpha = \{-0.8, 0.18 - 0.2\}$, under varying conditions of C_I and θ. Furthermore, only 210 points present a relative error greater than 10%, suggesting that the model's overall predictive capability is robust and effectively captures the rotational behavior.



(a) Values higher than μ.

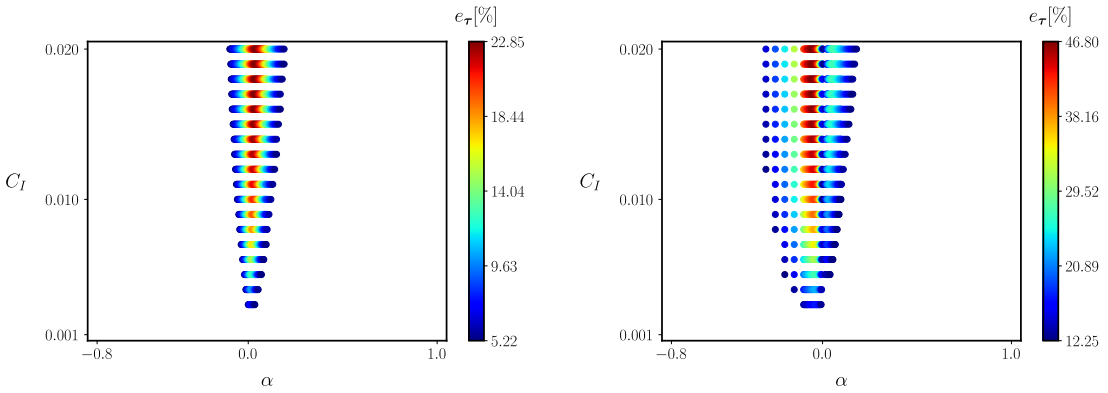
(b) Values higher than w_2.

Figure 5.6: Relative error distribution of $\boldsymbol{\tau}$.

Figure 5.7 illustrates the relative error distribution in the stress tensor across the parametric space, using the closure relation models. For the sake of clarity, data with relative error bellow $\mu_{\boldsymbol{\tau}}$ has been omitted. The analysis of Table 5.3 and Figure 5.7 allows to conclude that the Natural closure exhibits greater accuracy than the Hybrid closure at steady-state. The error distribution across the parametric space is broader for the Hybrid closure, evidencing reduced accuracy across a wider range of conditions. The relative error distribution for $\boldsymbol{\tau}$ reveals that the most problematic regions are near $\alpha \approx 0$, in the vicinity of simple-shear, with increasing relative error as the value of C_I rises. This behavior is anticipated for the Natural closure, as this relation was developed for the dilute state, where diffusion is absent. Additionally, as will be shown later, this issue is related to a misprediction of $\mathbf{A}_2$ in the same region. Overall, both conventional closure models considered yield predictions that are less accurate than the developed MLP model.

5.4.5 Prediction of second-order tensor

Given the significance of the second-order tensors in engineering computations, the prediction of $\mathbf{A}_2$ by retaining two modes of the RRAE demonstrates an average relative error of approximately

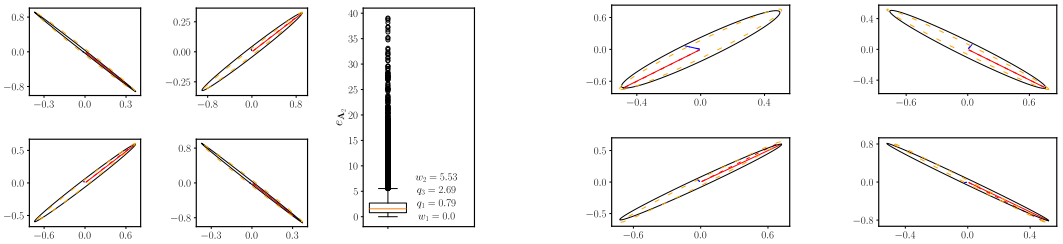


(a) Natural closure

(b) Hybrid closure

Figure 5.7: Relative error of $\boldsymbol{\tau}$ for each closure model. Data bellow average has been omitted.

2.1%. The box plot analysis for this metric shows an upper whisker at a relative error of 5.5%, beyond which points are considered as outliers. To underscore the quality of the predictions, Figure 5.8 illustrates two sets of four representative cases: those with relative errors near the mean and those at the upper-whisker boundary. Analysis of Figure 5.8a shows a reasonably accurate prediction relative to the true value of $\mathbf{A}_2$. Even in cases nearing the upper threshold, the SE representation closely aligns with the true SE representation of $\mathbf{A}_2$, confirming that the two leading modes of the SVD of the latent space sufficiently capture the orientation features provided by the second-order tensor. Furthermore, it is anticipated that incorporating the final mode will further refine these predictions and diminish the discrepancy error.



(a) Representative values near $\mu_{\mathbf{A}_2}$ and box plot analysis

(b) Representative values near w_2

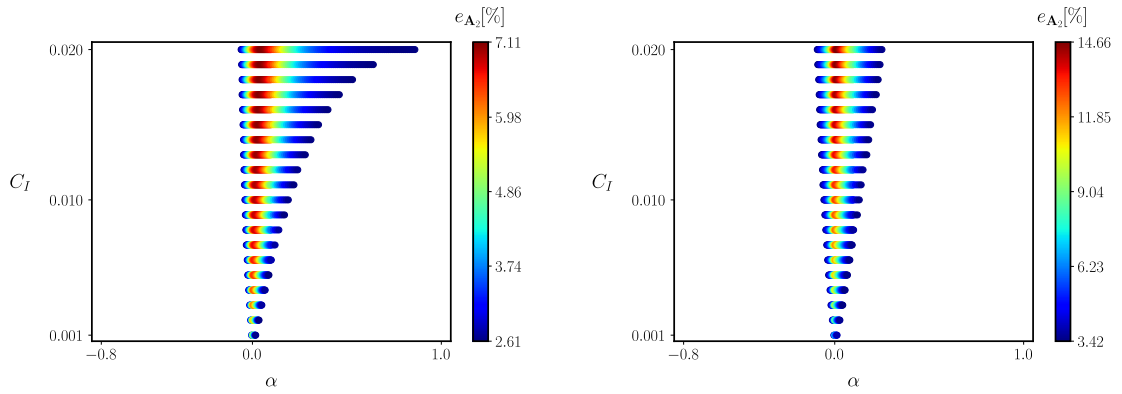
Figure 5.8: Prediction of $\mathbf{A}_2$

When utilizing the closure models, the relative error across the parametric space is depicted in Figure 5.9. Again, for clarity purposes, points with values below the average have been omitted.

Analysis of the figure reveals that the maximum obtained relative error is lower when employing the Natural closure. As noted in the stress prediction, the region exhibiting greater issues is situated near simple-shear, where the largest error in $\mathbf{A}_2$ is observed.

In comparison with the RRAE-MLP model, both conventional closures demonstrate a more constrained maximum error for the computed second-order tensor. The developed model exhibits an

overshoot when utilizing two leading modes to predict $\mathbf{A}_2$.

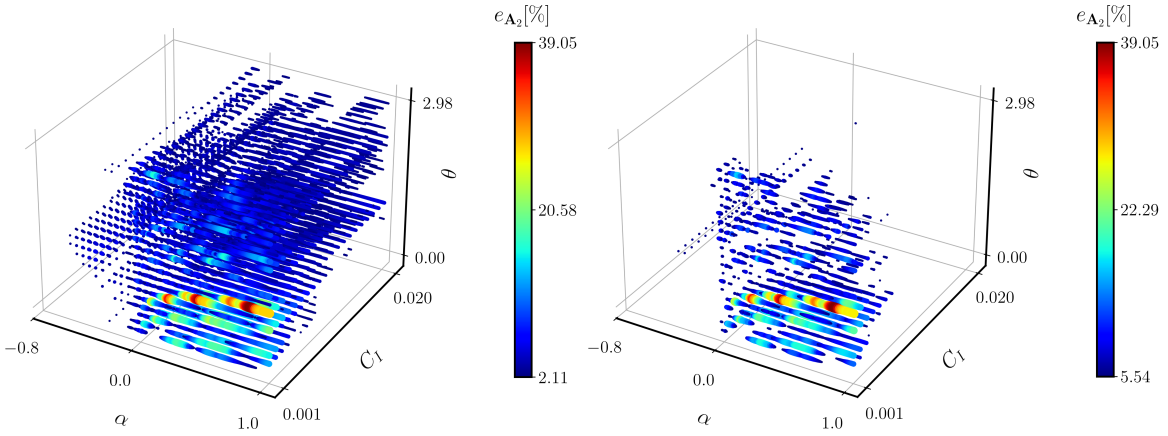


(a) Natural closure

(b) Hybrid closure

Figure 5.9: Relative errors of $\mathbf{A}_2$ for each closure model. Data bellow average has been omitted.

Figure 5.10 illustrates the distribution of relative error across the parametric space, omitting points with errors below the mean. Examination of the figure, in line with previous sections, indicates that only a small subset of conditions have a larger significant deviation from the exact values. For the second-order tensor, these conditions are highlighted in Figure 5.10b, and occur when C_I is set to its lowest value across varying values of α, with a notable concentration at $\alpha = [0.42, 0.5] \cup [0.77, 0.91]$.



(a) Values higher than μ.

(b) Values higher than w_2.

Figure 5.10: Relative errors of $\mathbf{A}_2$ over the parametric space

5.5 Predicting coefficient space

To broaden the functionality of the RRAE, we train a multi-layer perceptron network designed to map the parametric space variables to the coefficients of the latent space, denoted as c_i. This

parameter-to-coefficient surrogate enables to bypass the need to generate the ODFs that serve as inputs to the RRAE. Following the prediction of the coefficients, the RRAE proceeds with the reconstruction of the latent space and the decoder step, ultimately producing all quantities of interest. This streamlined framework offers a more efficient and convenient approach to exploit the RRAE technology. The overall procedure is depicted in Figure 5.11.

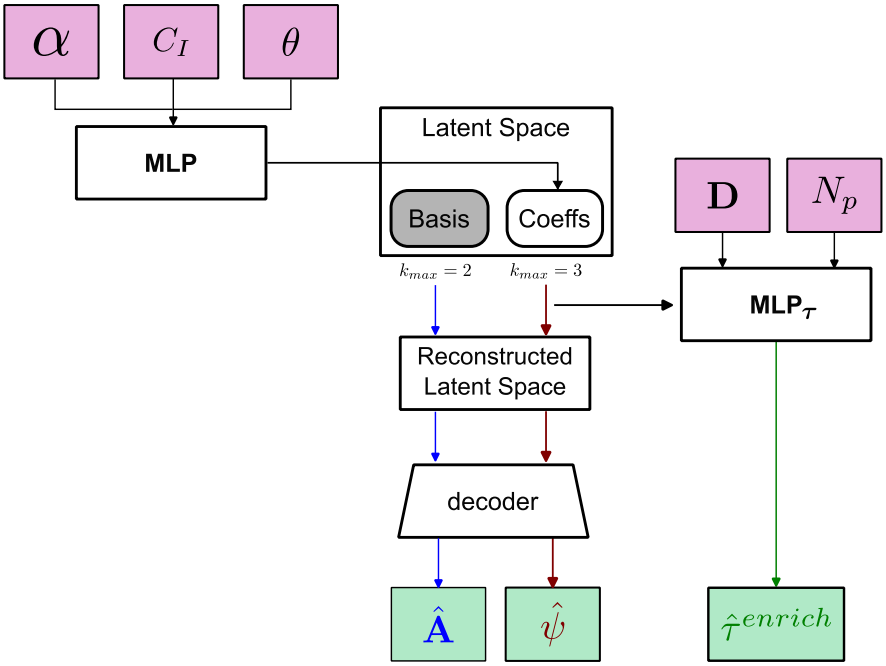


Figure 5.11: Coefficient space surrogate. Basis is already computed during the training process.

The MLP was composed of an input layer, three hidden layers with 30 neurons each, and an output layer. The ReLU activation function was employed in all layers except for the output layer, where the identity activation function was applied. No normalization or feature scaling was conducted on either the input variables or the latent space coefficients.

The dataset was divided, retaining $10\ 000$ points for testing. Training was performed in three stages, each lasting 750 epochs with a batch size of 20. The learning-rate decreased after each stage, following the sequence of $[1 \times 10^{-3}$, 1×10^{-4}, and $1 \times 10^{-5}]$. A relative $\mathcal{L}^2$-norm, similar to Equation (5.2) was utilized as the error metric for the loss function.

The relative errors in the coefficient space for both training and testing samples are presented in Table 5.4, accompanied by with a visual representation across the RRAE training parametric space in Figure 5.12.

Analysis of Table 5.4 reveals the presence of outliers in the parametric space, as evidenced by the extreme values. These outliers are illustrated in Figure 5.12.

For the selected network architecture and hyperparameters, the most significant errors occur at the lowest diffusion setting, within a limited range of rotations and α values. Despite these localized discrepancies, the average and standard deviation for both the training and testing sets remain low,

Table 5.4: Relative errors in the prediction of coefficients of latent space

	MLP train [%]	MLP test [%]
c	min/max: [0.03 - 56.88] $\mu = 2.54$ $\sigma = 2.99$	min/max: [0.07 - 40.91] $\mu = 2.59$ $\sigma = 3.17$

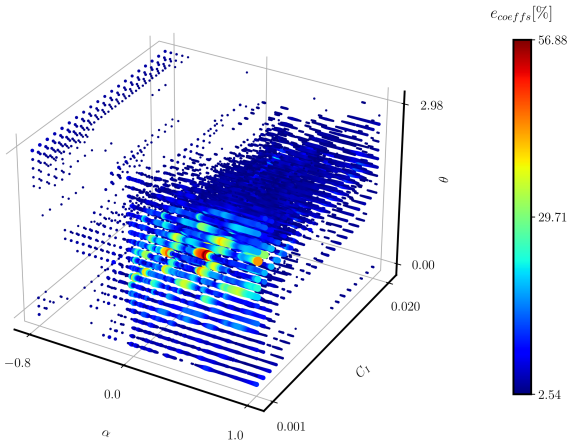


Figure 5.12: Coefficient space relative error. Values bellow average have been omitted.

suggesting an overall effective mapping to the coefficient space.

The integration of this parameter-to-coefficient mapping facilitates the reconstruction of the latent space, followed by the decoder step of the RRAE model, enabling the prediction of the steady-state quantities of interest. The results are summarized in Table 5.5, utilizing the same metrics reported in Section 5.4.

Table 5.5: Relative errors in dataset using predicted coefficients

	RRAE-MLP train [%]
$\mathbf{A}_2$	min/max: [0.00 - 22.44] $\mu = 2.69$ $\sigma = 2.18$
$\boldsymbol{\tau}$	min/max: [0.06 - 20.17] $\mu = 2.72$ $\sigma = 1.75$
ψ	min/max: [0.01 - 134.86] $\mu = 2.16$ $\sigma = 3.62$

The analysis of the table indicates that the results align closely with the baseline RRAE model taking the ODF as input, with errors that are only marginally larger. Some outliers are present, as evidenced by the extreme values across the parametric space. Nevertheless, the overall model performs well as reflected by the relatively error low mean value and standard deviation. The application of the predicted coefficients appears to induce a shift in the predicted ODF.

5.6 Conclusion

We have introduced a rank-reduction autoencoder combined with a multilayer perceptron that jointly compresses steady-state planar orientation distribution functions into a low-dimensional latent space, and predicts the suspension stress. Through the rank-reduction capability, the latent space was reconstructed using only three leading singular values. This reconstruction was utilized to derive the original input data together with a linear combination of the first two singular values to represent the second-order orientation tensor. Concurrently, the retained values from the singular valued decomposition, along with additional flow and suspension information, were given the MLP to establish a mapping to the unknown portion of the suspension stress.

The results indicate that the developed model performs well in predicting all quantities of interest across the selected parametric space, with the exception of certain localized regions. For a comparison, two state-of-the-art closure models were employed to predict stress and the second-order tensor at steady-state. Based on the obtained average relative error and box plot analysis, the developed model demonstrates superior accuracy under the examined conditions.

Although the current implementation of the Rank-Reduction Autoencoder-Multilayer Perceptron (RRAE-MLP) framework does not enforce equivariance, the data generated to train the model provided sufficient information to robustly capture this behavior.

This work demonstrates that this technology can practically be applied to orientation modeling, by embedding the rich meso-scale descriptors into a compact latent space representation, retaining only a few singular values, thereby extracting key orientation features. Despite lacking direct physical meaning, the latent space serves as an efficient encoding that the decoder can utilize to reconstruct the full orientation distribution function.

There are several opportunities for improvement within this methodology. First, use an architecture able to enforce the velocity gradient rotation–equivariant behavior to reduce the amount of training data required and enhance generalization. Subsequently, the developed framework should be extended to handle transient ODFs evolutions. Finally, adapting the RRAE-MLP to handle full three-dimensional ODFs through more advanced machine learning architectures, such as convolutional networks, to encode the meso-scale descriptors.

Appendix

Availability of data and material

The data that support the findings of this study are openly available at: `https://github.com/Computational-Rheology/FOM_RRAE_planar`

References

[1] Suresh G. Advani and Charles L. Tucker III. The use of tensors to describe and predict fiber orientation in short fiber composites. *Journal of Rheology*, 31(8):751–784, 11 1987.

[2] Jin Wang, John F. O'Gara, and Charles L. Tucker III. An objective model for slow orientation kinetics in concentrated fiber suspensions: Theory and rheological evidence. *Journal of Rheology*, 52(5):1179–1200, 09 2008.

[3] Jay H. Phelps and Charles L. Tucker III. An anisotropic rotary diffusion model for fiber orientation in short- and long-fiber thermoplastics. *Journal of Non-Newtonian Fluid Mechanics*, 156(3):165–176, 2009.

[4] Huan-Chang Tseng, Rong-Yeu Chang, and Chia-Hsiang Hsu. An objective tensor to predict anisotropic fiber orientation in concentrated suspensions. *Journal of Rheology*, 60(2):215–224, 03 2016.

[5] Huan-Chang Tseng, Rong-Yeu Chang, and Chia-Hsiang Hsu. The use of principal spatial tensor to predict anisotropic fiber orientation in concentrated fiber suspensions. *Journal of Rheology*, 62(1):313–320, 01 2018.

[6] Adrien Scheuer, Amine Ammar, Emmanuelle Abisset-Chavanne, Elías Cueto, Francisco Chinesta, Roland Keunings, and Suresh G. Advani. Data-driven upscaling of orientation kinematics in suspensions of rigid fibres. *Computer Modeling in Engineering & Sciences*, 117(3):367–386, 2018.

[7] Susanne Katrin Kugler, Argha Protim Dey, Sandra Saad, Camilo Cruz, Armin Kech, and Tim Osswald. A flow-dependent fiber orientation model. *Journal of Composites Science*, 4(3), 2020.

[8] Jin Wang and Xiaoshi Jin. Comparison of recent fiber orientation models in autodesk moldflow insight simulations with measured fiber orientation data. In *Proceedings of the polymer processing society 26th annual meeting*, July 2010.

[9] Huan-Chang Tseng, Rong-Yeu Chang, and Chia-Hsiang Hsu. Comparison of recent fiber orientation models in injection molding simulation of fiber-reinforced composites. *Journal of Thermoplastic Composite Materials*, 33(1):35–52, 2020.

[10] Alexander Kriwet and Markus Stommel. The impact of fiber orientation on structural dynamics of short-fiber reinforced, thermoplastic components—a comparison of simulative and experimental investigations. *Journal of Composites Science*, 6(4), 2022.

[11] Anthony J. Favaloro and Charles L. Tucker III. Analysis of anisotropic rotary diffusion models for fiber orientation. *Composites Part A: Applied Science and Manufacturing*, 126:105605, 2019.

[12] Suresh G. Advani and Charles L. Tucker III. *Closure Approximations for Three-Dimensional Orientation Structure Tensors*, pages 13–15. Springer Netherlands, Dordrecht, 1990.

[13] Kevin Breuer, Markus Stommel, and Wolfgang Korte. Analysis and evaluation of fiber orientation reconstruction methods. *Journal of Composites Science*, 3(3), 2019.

[14] Susanne Katrin Kugler, Armin Kech, Camilo Cruz, and Tim Osswald. Fiber orientation predictions—a review of existing models. *Journal of Composites Science*, 4(2), 2020.

[15] Bruno Ramoa, Chady Ghnatios, João Miguel Nóbrega, and Francisco Chinesta. Physics-informed data-driven closure relation for dilute short fiber suspensions. *Adv. Model. Simul. Eng. Sci.*, 12(1), March 2025.

[16] Steven M. Dinh and Robert C. Armstrong. A rheological equation of state for semiconcentrated fiber suspensions. *Journal of Rheology*, 28(3):207–227, 06 1984.

[17] Arif Hussain, Amir Hosein Sakhaei, and Mahmood Shafiee. Machine learning-based constitutive modelling for material non-linearity: A review. *Mechanics of Advanced Materials and Structures*, 0(0):1–19, 2024.

[18] Jad Mounayer, Sebastian Rodriguez, Chady Ghnatios, Charbel Farhat, and Francisco Chinesta. Rank reduction autoencoders, 2025.

[19] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression, 2017.

[20] Min Chen, Xiaobo Shi, Yin Zhang, Di Wu, and Mohsen Guizani. Deep feature learning for medical image analysis with convolutional autoencoder neural network. *IEEE Transactions on Big Data*, 7(4):750–758, 2021.

[21] Shengli Jiang, Adji Bousso Dieng, and Michael A Webb. Property-guided generation of complex polymer topologies using variational autoencoders. *Npj Comput. Mater.*, 10(1), June 2024.

[22] Xiaolong He, Qizhi He, and Jiun-Shyan Chen. Deep autoencoders for physics-constrained data-driven nonlinear materials modeling. *Computer Methods in Applied Mechanics and Engineering*, 385:114034, 2021.

[23] Charles L. Tucker III. Planar fiber orientation: Jeffery, non-orthotropic closures, and reconstructing distribution functions. *Journal of Non-Newtonian Fluid Mechanics*, 310:104939, 2022.

[24] Fransisco Folgar and Charles L. Tucker III. Orientation behavior of fibers in concentrated suspensions. *Journal of Reinforced Plastics and Composites*, 3(2):98–119, 1984.

[25] Charles L. Tucker III. *Fundamentals of Fiber Orientation: Description, Measurement and Prediction*. Carl Hanser Verlag GmbH, 2022.

[26] J. Férec, M. Heniche, M.C. Heuzey, G. Ausias, and P.J. Carreau. Numerical solution of the fokker–planck equation for fiber suspensions: Application to the folgar–tucker–lipscomb model. *Journal of Non-Newtonian Fluid Mechanics*, 155(1):20–29, 2008.

[27] Charles L. Tucker III. Fiber orientation tools. `https://github.com/charlestucker3/Fiber-Orientation-Tools`, 2022.

[28] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[29] David A. Jack, Bryan Schache, and Douglas E. Smith. Neural network-based closure for modeling short-fiber suspensions. *Polymer Composites*, 31(7):1125–1141, 2010.

[30] Charles L. Tucker III. Flow regimes for fiber suspensions in narrow gaps. *Journal of Non-Newtonian Fluid Mechanics*, 39(3):239–268, 1991.

[31] N. Phan-Thien, X.-J. Fan, R.I. Tanner, and R. Zheng. Folgar–tucker constant for a fibre suspension in a newtonian fluid. *Journal of Non-Newtonian Fluid Mechanics*, 103(2):251–260, 2002.

[32] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[33] Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James S. Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients, 2020.

[34] DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020.

[35] Scipy. Wasserstein distance. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wasserstein_distance.html`, 2025. Accessed: 20-08-2025.

[36] Gordon L. Kindlmann and Thomas Schultz. Superquadric Glyphs for Symmetric Second-Order Tensors . *IEEE Transactions on Visualization & Computer Graphics*, 16(06):1595–1604, 2010.

Chapter 6 A semi-automatic approach to assess the convergence order in OpenFOAM®

Abstract:

Code verification is an intricate but crucial part of numerical code development. Due to the complexity of the partial differential equations, an analytical solution might not exist. In those situations and aiming at proving that the code is solving appropriately the governing equations, the method of manufactured solutions (MMS) is a powerful tool. In this method, a source term is derived to enforce the solution to a predetermined function. By performing a mesh refinement study, one can verify if the code is correctly solving the desired equations. In this work, a methodology that allows the automation of the MMS within the OpenFOAM® framework is proposed. The developed computational framework comprises a set of tools prepared, in an open-source environment, for the symbolic computation of the associated source term, and to generate the code required for its implementation as well as appropriate boundary conditions and functions to calculate the error norms.

Keywords: Solver verification; Method of Manufactured Solutions; Symbolic computation; OpenFOAM® coded framework

This chapter was adapted from Ramoa, B., Costa, R., Chinesta, F., & Nóbrega, J. M. (2022). A Semi-Automatic Approach Based on the Method of Manufactured Solutions to Assess the Convergence Order in OpenFOAM. OpenFOAM® Journal, 2, 148-165. https://doi.org/10.51560/ofj.v2.75

6.1 Introduction

Code verification is an intricate but crucial part of numerical code development [1, 2]. In computational modeling, systems of partial differential equations (PDE) equipped with appropriate boundary conditions are used to model several physical phenomena, such as fluid dynamics, structural mechanics, conjugated heat transfer, combustion, transport in porous media, etc. In that context, verification procedures aim at demonstrating that the continuum governing equations are solved correctly, and that the computed solution tends asymptotically to the exact solution as finer meshes and smaller time-step sizes, are applied. In that regard, the convergence order of the method measures the rate at which the error decreases with mesh and time-step refinement.

Several methodologies of verification can be employed to demonstrate that the code correctly solves the physical model. The classical approach consists in deriving an analytical (exact) solution for the governing equations, which is usually only feasible for problems with reduced dimension (1D or 2D) and simple geometries. For some physical phenomena, exact solutions might not be available even for simplified setups. This is usually known as the method of exact solutions and has been used extensively in the context of OpenFOAM® for solver verification [3]. However, for real-world problems, the models to solve often contain non-linear terms, couplings, variable coefficients, and complex geometries [4], which hinders or makes it impossible to derive analytical solutions. Moreover, depending on the mathematical techniques employed for deriving the exact solution, the use of infinite sums, special functions or non-trivial integrals will raise issues, such as where to truncate the summation, what integration scheme should be used, and the possible existence of singularities in the solution [4].

The method of manufactured solutions (MMS) is an alternative methodology that is more practical and interesting for code verification in complex problems [4–6]. The method consists in providing an analytical solution for the unknowns of the problem as a function of space and time. Then, specific source terms for each governing equation are determined by applying the corresponding differential operators to the predefined solution. Plugging the computed source terms into the original PDE(s) will result in a new set of equations whose exact solution is the manufactured solution. The convergence order of the method can then be assessed by performing a series of tests, through either a mesh or time-step refinement, or a combination of both.

In most situations, the manufactured solution should be smooth and regular in space and time (for unsteady problems), non-trivial, and allow the calculation of all derivatives comprised in the PDE(s) (e.g. cross-derivatives). Additionally, the best type of manufactured solutions should be described with simple functions, such as trigonometric, exponential or high degree polynomials. Detailed guidelines for creating manufactured solutions can be found in Salari and Knupp [4].

In the context of OpenFOAM®, the MMS has been employed to assess different solvers, covering several areas, such as computational fluid dynamics [7–10], solid mechanics [11],

fluid-structure interaction [12], heat transfer [13], nuclear engineering [14], reacting flows [15], among others. However, in all the referred works, the MMS is used to assess the developed solvers, but no details are provided to guide users interested in extending the approach to other cases. Moreover, the manipulation of the complex analytical expressions associated even with simple cases, and the implementation of required source terms and boundary conditions is often a complex, labour-intensive and error-prone task.

In the present work, a semi-automatic approach to undertake code verification in OpenFOAM® through the MMS is proposed and illustrated. The approach allows a straightforward extension to different case studies and solvers. It resorts to `SymPy` [16] v1.8, an open-source symbolic computation module from `Python` programming language, and OpenFOAM®'v1912 `fvOptions` and `coded` functionality, which allows a high degree of flexibility for a swift, and automatic, computation of source terms and boundary conditions.

The remaining paper is organized as follows. Section 6.2 introduces the MMS and the computation of associated error norms and convergence orders. In Section 6.3, the semi-automatic approach for implementing the MMS in OpenFOAM® is presented. The subsequent section, Section 6.4, a number of case studies involving the `laplacianFoam` [17] and `simpleFoam` [18] solvers are carried out, using structured meshes, to demonstrate the application of the proposed semi-automatic approach. In Section 6.5, the results obtained for the mentioned case studies are presented and discussed. Finally, in Section 6.6, the main conclusions of the present work are drawn.

6.2 Methodology

6.2.1 Unsteady heat transfer problem

In the present section, the methodology of the MMS is described in more detail and illustrated for the two-dimensional unsteady heat equation, considering only conduction. The governing equation is solved for a spatial domain $(x, y) \in \Omega$ with boundary $\Gamma = \Gamma^{\mathrm{D}} \cup \Gamma^{\mathrm{N}}$, and time range $t \in \left(0, t^{\mathrm{F}}\right]$, equipped with appropriate boundary and initial conditions, and reads:

$$\frac{\partial T}{\partial t} - \nabla \cdot (D_T \nabla T) = S_{\mathrm{T}}, \qquad \text{in } \Omega \times \left(0, t^{\mathrm{F}}\right], \tag{6.1}$$

$$T = g^{\mathrm{D}}, \qquad \text{in } \Gamma^{\mathrm{D}} \times \left(0, t^{\mathrm{F}}\right], \tag{6.2}$$

$$\nabla T \cdot \mathbf{n} = g^{\mathrm{N}}, \qquad \text{in } \Gamma^{\mathrm{N}} \times \left(0, t^{\mathrm{F}}\right], \tag{6.3}$$

$$T(x, y, t = 0) = T^0, \qquad \text{in } \Omega, \tag{6.4}$$

where:

- $T \equiv T(x, y, t)$ is the unknown temperature distribution,

- $D_\mathrm{T} \equiv D_\mathrm{T}(x, y, t)$ is the thermal diffusivity coefficient,
- $S_\mathrm{T} \equiv S_\mathrm{T}(x, y, t)$ is the heat source term (sinks have a negative value),
- $g^\mathrm{D} \equiv g^\mathrm{D}(x, y, t)$ is a prescribed boundary temperature value,
- $g^\mathrm{N} \equiv g^\mathrm{N}(x, y, t)$ is a prescribed boundary temperature gradient normal to the boundary,
- $T^0 \equiv T^0(x, y)$ is the temperature distribution at time $t = 0$, and
- $\mathbf{n} \equiv \mathbf{n}(x, y) \equiv \left(n_x(x, y),\, n_y(x, y)\right)$ is the unit outward vector normal to the boundary.

In the above notation, the superscripts `F`, `D` and `N` stand for "Final", "Dirichlet" and "Neumann", respectively.

6.2.2 Method of manufactured solutions

Consider that some function $T_\mathrm{ex} \equiv T_\mathrm{ex}(x, y, z, t)$ is the manufactured solution of the unsteady heat transfer problem, which must be regular and smooth in space and time. For instance, trigonometric functions, such as the sine and cosine, tend to be good choices, since they are continuous and infinitely differentiable. Following the methodology proposed in [4], a source term function $S_\mathrm{T}(x, y, z, t)$ for the unsteady heat equation is determined as $S_\mathrm{T} = \partial T_\mathrm{ex}/\partial t - \nabla \cdot (D_\mathrm{T} \nabla T_\mathrm{ex})$. Then, the unsteady heat equation, Equation (6.1), is solved, considering appropriate discretization method(s) for the unknown variable, $T(x, y, z, t)$, and including the source term and boundary conditions determined with the manufactured solution, $T_\mathrm{ex}(x, y, z, t)$. Finally, having an approximate discrete solution for the unknown temperature function, the corresponding errors can be computed, since the exact solution for this specific problem was previously manufactured and, therefore, is known.

Example 1 *The application of the MMS to the unsteady heat transfer problem is hereafter demonstrated with a simple example. Assume a constant value for D_T and a manufactured solution for a 2D case study given as:*

$$T_{ex}(x, y, t) = 150\left(\cos\left(x^2 + y^2 + \omega t\right) + 1.5\right). \tag{6.1}$$

As previously presented, the corresponding heat source term, $S_T(x, y, t)$, is obtained by replacing, in the governing PDE (6.1)*, the unknown temperature distribution, $T(x, y, t)$, with the manufactured counterpart, $T_{ex}(x, y, t)$, which yields:*

$$
\begin{aligned}
S_T(x,y,t) &= \frac{\partial T_{ex}(x,y,t)}{\partial t} - \nabla \cdot (D_T \nabla T_{ex}(x,y,t)) \\
&= 600 D_T x^2 \cos\left(\omega t + x^2 + y^2\right) + 600 D_T y^2 \cos\left(\omega t + x^2 + y^2\right) \\
&\quad + 600 D_T \sin\left(\omega t + x^2 + y^2\right) - 150\omega \sin\left(\omega t + x^2 + y^2\right).
\end{aligned} \tag{6.2}
$$

The problem is equipped with the appropriate boundary conditions, assuming either a prescribed boundary temperature (Dirichlet boundary condition) or a prescribed temperature normal gradient (Neumann boundary condition). In any case, the corresponding functions to be imposed in Equations (6.2) *and* (6.3)*, are computed based on the manufactured solution,* $T_{ex}(x,y,t)$*. For this example, the Dirichlet boundary condition function is determined as:*

$$
g^D(x,y,t) = T_{ex}(x,y,t) = 150\left(\cos\left(x^2 + y^2 + \omega t\right) + 1.5\right), \tag{6.3}
$$

while the Neumann boundary condition function is given by:

$$
\begin{aligned}
g^N(x,y,t) &= \nabla T_{ex}(x,y,t) \cdot \mathbf{n} \\
&= \begin{bmatrix} -300x \sin\left(\omega t + x^2 + y^2\right) \\ -300y \sin\left(\omega t + x^2 + y^2\right) \end{bmatrix} \cdot \begin{bmatrix} n_x \\ n_y \end{bmatrix} \\
&= -300 \sin\left(\omega t + x^2 + y^2\right)\left(n_x x + n_y y\right).
\end{aligned} \tag{6.4}
$$

Other boundary conditions, such as the Robin boundary condition, can be prescribed following the same procedure. The initial condition is simply set as $T^0(x,y) = T_{ex}(x,y,t=0)$*. In practice, different combinations of boundary conditions can be selected for the boundaries of the domain.*

Finally, solving the governing PDE, Equation (6.1)*, using the previously determined source term (Equation* (6.2)*), boundary (Equation* (6.3) *and/or* (6.4)*), and initial values, the exact solution of the unsteady heat transfer problem is the manufactured solution, Equation* (6.1)*.*

6.2.3 Errors and convergence orders

Having a manufactured solution for the problem under study, the employed numerical techniques (discretization method, solution algorithm, etc.) can be verified by determining the accuracy of the computed approximate solution. For that purpose, consider that vector $\overline{\mathbf{T}}^n$ gathers the exact piece-wise solution determined from the manufactured solution, Equation (6.1), with a given mesh and at a specific time $t = t^n$. On the other side, vector $\mathbf{T}^{n,*}$ gathers the approximate values obtained with the numerical code in the same mesh and at the same instant of time, considering the appropriate source term, boundary and initial conditions. In the context of the finite volume method, the exact value for a given cell C_i at time $t = t^n$, denoted as $\overline{T}_i^n$, corresponds to the cell mean-value of the manufactured solution,

that is:

$$\overline{T}_i^n = \iint_{C_i} T_{\text{ex}}(x, y, t^n)\, \mathrm{d}x\, \mathrm{d}y, \tag{6.1}$$

whereas the corresponding approximate value, denoted as $T_i^{n,*}$, is defined at the cell mass center. Then, a global measure of the error can be calculated, where the L^1-, L^2- and L^∞-norms are usually chosen, for which the associated errors at time t^n are denoted as E^1, E^2 and E^∞, and are defined as:

$$E^1 = \frac{\sum_{i=1}^{N_\text{C}} \left|\overline{T}_i^n - T_i^{n,*}\right| V_i}{\sum_{i=1}^{N_\text{C}} V_i}, \tag{6.2}$$

$$E^2 = \sqrt{\frac{\sum_{i=1}^{N_\text{C}} \left(\overline{T}_i^n - T_i^{n,*}\right)^2 V_i}{\sum_{i=1}^{N_\text{C}} V_i}}, \tag{6.3}$$

$$E^\infty = \max_{i=1}^{N_\text{C}} \left|\overline{T}_i^n - T_i^{n,*}\right|, \tag{6.4}$$

where:

- N_C is the total number of cells in the mesh, and
- V_i is the volume of cell C_i.

In the L^1-norm, the absolute individual errors are scaled with the corresponding cell volumes and, therefore, all the cells have a comparable contribution to the error norm for reasonably uniform meshes. On the other side, in the L^2-norm, also known as the Euclidean norm, the weight of the cells with larger errors is more pronounced when compared to the L^1-norm. Finally, the L^∞-norm represents the largest error magnitude in the whole mesh. The comparison of the errors in the L^1- or L^2-norm with the corresponding error in the L^∞-norm is also interesting as it provides an insight into the error distribution without visualizing it. That is, if the errors in the L^1- or L^2-norm is comparable to the corresponding error in the L^∞-norm, then the individual errors have a relatively even distribution. On the contrary, when the ratio between these error norms becomes more significant, larger errors do exist for specific cells in the mesh, which indicates that a careful analysis might be necessary.

The errors of the computed approximate solution obtained with just one mesh (or time-step) are not sufficient to conclude whether the method is correctly implemented. Indeed, provided that the

method is consistent, the approximate (numerical) solution should converge to the exact counterpart, while the mesh characteristic or time-step sizes decrease. In that case, the rate at which the approximate solution error decreases under mesh or time-step refinement is called convergence order. More specifically, for steady-state problems or unsteady problems at a certain instant of time, the error in some norm is given as:

$$E = Ch^P + H.O.T., \tag{6.5}$$

where:

- C is a constant independent of the mesh size,
- h is the mesh characteristic size, a representative measure of the size of the elements in the mesh, such that for the particular case of structured uniform meshes it corresponds to the length of the edges,
- P is a constant that determines the rate at which the error changes with h, the so-called convergence order, and
- $H.O.T.$ represents the higher order terms that are not explicitly defined.

The contribution of $H.O.T.$ can be usually neglected and, therefore, the error in some norm can be given as $E = Ch^P$, that is, proportional to h^P. In that case, as the mesh characteristic size decreases, the error is also expected to decrease with h at a rate corresponding to P. For unsteady problems, the same premises hold and, therefore, the error in some norm is also a function of the employed time-step size, Δt.

In practice, for steady-state problems, the convergence order of the method can be determined by solving the same test case under the same conditions with successively finer meshes. In that case the ratio between the characteristic size of two consecutive meshes, r_h (referred to as coarser and finer), is given as:

$$r_h = \frac{h_{\text{coarser}}}{h_{\text{finer}}} \approx \left(\frac{N_{\text{finer}}}{N_{\text{coarser}}} \right)^{\frac{1}{D}}, \tag{6.6}$$

where:

- h_{coarser} is the characteristic size of the coarser mesh,
- h_{finer} is the characteristic size of the finer mesh,
- N_{coarser} is the number of cells in the coarser mesh,
- N_{finer} is the number of cells in the finer mesh, and
- D is the problem dimension ($D = 1, 2, 3$ for a 1D, 2D and 3D problems, respectively).

Finally, the convergence orders for the errors in the L^1-, L^2- and L^∞-norms, denoted as O^1, O^2 and O^∞, respectively, are given as:

$$O^1 = \frac{\ln\left(E^1_{\text{coarser}}/E^1_{\text{finer}}\right)}{\ln\left(r_h\right)}, \tag{6.7}$$

$$O^2 = \frac{\ln\left(E^2_{\text{coarser}}/E^2_{\text{finer}}\right)}{\ln\left(r_h\right)}, \tag{6.8}$$

$$O^\infty = \frac{\ln\left(E^\infty_{\text{coarser}}/E^\infty_{\text{finer}}\right)}{\ln\left(r_h\right)}, \tag{6.9}$$

where E^1_{coarser}, E^2_{coarser} and $E^\infty_{\text{coarser}}$ are the the L^1-, L^2- and L^∞-norms errors for the coarser mesh and E^1_{finer}, E^2_{finer} and E^∞_{finer} are the same error norms for the finer mesh.

The same analysis can be performed to assess the convergence order of the method under mesh refinement for unsteady problems, employing the previous procedure for a fixed time-step size. However, in that case, a sufficiently small time-step size must be carefully chosen, such that the time discretization error does not exceed the space discretization counterpart. To determine if the chosen time-step size is appropriate, the same simulations can be replicated with a slightly smaller time-step size, but with the same successively finer meshes. If the computed solution error for any of the meshes is significantly different than previously for the same mesh, the time discretization error is still predominant. In that case, the verification procedure must be repeated until no significant changes are observed in the computed solution errors from employing a smaller time-step size. This is accomplished by selecting the time-step below which the reported significant digits for the error remain unchanged. It is important to notice that since the time and space discretization errors decrease under mesh and time refinement, respectively, larger time-step sizes can be used for coarser meshes, without affecting the obtained conclusions. Accordingly, as smaller mesh characteristic sizes are considered, decreasing progressively the time-step size might be required to avoid the influence of the time discretization error on the results obtained. Consequently, identifying an appropriate time-step size for each mesh refinement level might be more difficult to employ but can provide significant computational savings in computationally demanding test cases.

On the other side, to assess the convergence order under time-step refinement, the same procedure can be employed, considering successively smaller time-step sizes for a fixed mesh characteristic size. In that case, Equations (6.7) to (6.9) for the convergence order in time must consider the ratio between two successive time-step sizes, $r_{\Delta T}$. As before, a sufficiently small mesh characteristic size must be carefully chosen such that the space discretization error does not exceed the time discretization error. To determine the appropriate mesh characteristic size, the same simulations can be replicated with a finer mesh until no significant changes are observed in the computed solution errors. Alternatively, decreasing simultaneously both the mesh characteristic and the time-step sizes can be computationally more efficient, although it is generally more difficult to employ.

6.3 Semi-automatic approach

Based on a selected manufactured solution, the implementation of the MMS approach in OpenFOAM® requires the calculation of: (i) the associated distributed source term, (ii) the associated Dirichlet and/or Neumann boundary condition functions, and (iii) the `coded functionObject` for the computation of the error norms. Given the arbitrary complexity of choosing a non-trivial manufactured solution, the programming of the associated source term and boundary condition functions can become a cumbersome and error-prone task. In that regard, a `Python` package (referred to as `pyMMSFoam`) was created to perform the necessary mathematical differentiation based on the `SymPy` v1.8 module [16], a rich-featured library for symbolic computation written in `Python`. The package converts the generated mathematical expressions to C-syntax so that they can be copied directly into an OpenFOAM® case setup files. In addition, the `SymPy`'s built-in common sub-expression elimination (CSE) routine [19] is used to simplify the mathematical expressions and, therefore, optimize the generated code.

The semi-automatic approach consists, firstly, in generating the required inputs for the implementation of the MMS approach in OpenFOAM®, which can be easily performed with the developed package. Then, the test case setup is prepared in OpenFOAM® and consecutive simulations are performed to determine the computed solution errors and convergence orders. The flowchart of the proposed semi-automatic approach based on the MMS in OpenFOAM® is illustrated in Figure 6.1 and is described in more detail hereafter. For the sake of simplicity, consider a 2D unsteady heat transfer equation, given by Equation (6.1), with $\omega = 0.1$ and $D_\mathrm{T} = 10^{-3}$ m^2/s. This will also be used for the case study of the `laplacianFoam` solver in Section 6.4.1.

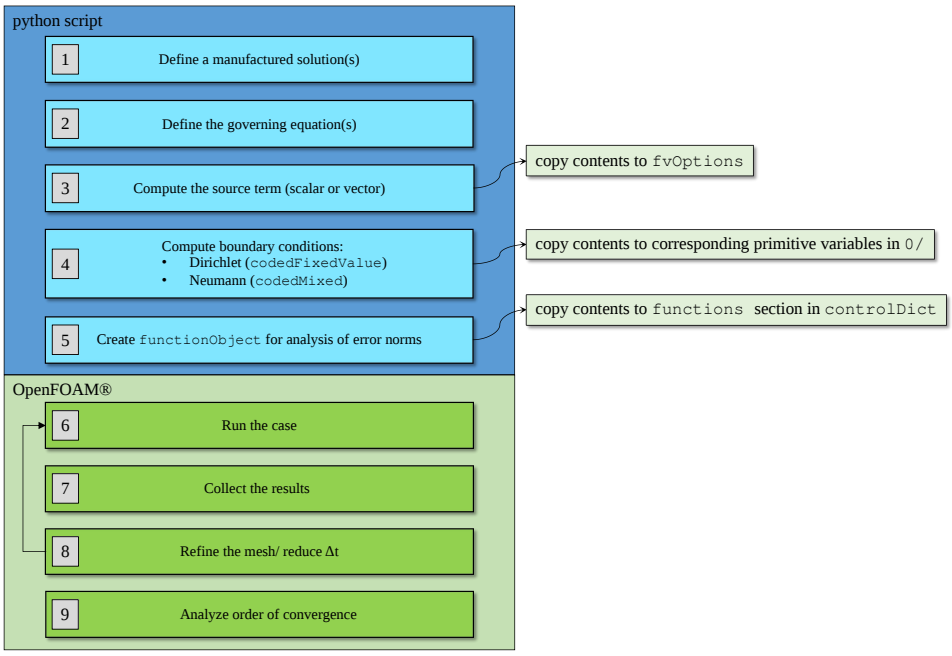


Figure 6.1: Flow chart of the semi-automatic approach based on the MMS in OpenFOAM®.

In Step 1 (see Figure 6.1), the manufactured solution has to be defined, and the user can rely on the SymPy's built-in functionality, such as trigonometric and exponential functions. Listing 6.1 illustrates the code used for the 2D unsteady heat transfer case (Equation (6.1)), with the above referred parameters.

From lines 1 to 3, the `pyMMSFoam` package is being loaded with the alias `mms`, the Cartesian space symbolic variables `x`, `y` and `z`, the temporal symbolic variable `t`, and the `cos` function from SymPy are being imported from their respective modules. Notice that the symbolic variable `z` is not required for this specific manufactured solution, but it is shown for the sake of completeness and easy adaptation to other cases. In lines 5 and 7, the value of diffusion coefficient and the manufactured solution, Equation (6.1), are declared.

```
1  import pyMMSFoam as mms
2  from pyMMSFoam import x,y,z,t
3  from sympy import cos
4
5  DT = 1e-3
6
7  T = 150*(cos(x*x + y*y + 0.1*t) + 1.5)
8
9  S_T = mms.ddt(T) - mms.laplacian(T, DT)
10
11 mms.generateFvOptions(S_T, "sourceTerm", "T")
12 mms.generateDirichletBoundaries(T, "T")
13 mms.generateNeumannBoundaries(T, "T")
14 mms.generateFunctionObject(T, "T")
```

Listing 6.1: `laplacianFoam` MMS script.

The differential operators available in `pyMMSFoam` are the following: time derivative (for scalar and vector quantities), divergence (for vector and tensor quantities), gradient (for scalar and vector quantities) and Laplacian (for scalar quantities). Moreover, `pyMMSFoam` interprets a symbolic expression as a scalar quantity, a matrix of shape (3 x 1) as a vector and a matrix of shape (3 x 3) as a tensor.

For Steps 2 to 5 (see Figure 6.1), the generation of the input codes for the the MMS implementation in OpenFOAM® is performed. In the second part of Listing 6.1, the source term, Dirichlet and Neumann boundary conditions are symbolically created. Moreover, a `coded functionObject` is also generated to compute the approximate solution error using different norms (Equations 6.2, 6.3 and 6.4) .

In line 9 of Listing 6.1, the associated source term is computed from the governing equation for the unsteady heat transfer problem, Equation (6.1), that includes the time derivative and Laplacian differential operators. The command presented in line 11 generates a `fvOptions` dictionary, by running the function `generateFvOptions`, whose arguments are the computed source term and

two strings, the names of the source term and the field where the source term should be applied (T for the present problem). The implemented function will check whether the source term is a scalar or a vector quantity and write a `scalarCodedSource` or a `vectorCodedSource` `fvOptions` dictionary, respectively. The output for this example is presented in Listing 6.2.

```
sourceTerm
{
   type scalarCodedSource;
   selectionMode all;
   fields (T);
   name sourceTerm_2;

   codeInclude
   #{
   #};
   codeCorrect
   #{
   #};
   codeConstrain
   #{
   #};
   codeAddSup
   #{
            const scalarField& V = mesh_.V();
            const volVectorField& C = mesh().C();
            scalarField& TSource= eqn.source();

            const Time& time = mesh().time();
            const scalar t = time.value();

            forAll(TSource, cellI)
            {
                  const scalar x = C[cellI].x();
                  const scalar y = C[cellI].y();

                  const scalar tmp0 = x*x;
                  const scalar tmp1 = y*y;
                  const scalar tmp2 = 0.10*t + tmp0 + tmp1;
                  const scalar tmp3 = 0.6*Foam::cos(tmp2);
                  const scalar solution_T = tmp0*tmp3 + tmp1*tmp3 -
                  14.4*Foam::sin(tmp2);

                  TSource[cellI] -= V[cellI]*(solution_T);
            };
   #};
}
```

Listing 6.2: `fvOptions` dictionary generated for the 2D unsteady heat transfer equation (`laplacianFoam` solver).

In lines 12 and 13 of Listing 6.1 Dirichlet and Neumann boundary conditions are generated using a `codedFixedValue` and `codedMixed` boundary condition, respectively. The arguments of these functions are the symbolic variable that represents the manufactured solution and a string with the field name. Analogously to what was described for function `generateFvOptions`, the tool will check whether the input is a scalar or vector quantity, and generate the output accordingly. For this example, the output for each type of boundary condition is illustrated in Listings 6.3 and 6.4. Notice that the name of the patches should be changed afterwards from `"(patch1|patch2|patch3)"` to the ones defined in the test case setup.

```
1  "(patch1|patch2|patch3)"
2  {
3     type codedFixedValue;
4     value uniform 0;
5
6     name T_dirichlet;
7
8     code
9     #{
10       const fvPatch& boundaryPatch = patch();
11
12       const vectorField& Cf = boundaryPatch.Cf();
13
14       scalarField& field = *this;
15
16       const scalar t = this->db().time().value();
17
18       forAll(Cf, faceI)
19       {
20          const scalar x = Cf[faceI].x();
21          const scalar y = Cf[faceI].y();
22          const scalar T = 225.0 + 150*Foam::cos(0.10*t + x*x + y*y);
23
24          field[faceI] = T ;
25       }
26    #};
27 }
```

Listing 6.3: Dirichlet boundary condition generated for the 2D unsteady heat transfer equation (`laplacianFoam` solver).

Finally, in line 14 of Listing 6.1, a `coded functionObject` is created to calculate the computed solution error using the norms previously defined (Equations 6.2, 6.3 and 6.4). The function arguments are the variable that represents the manufactured solution and a string with the field variable name. The output of this function for this illustrative example is provided in Listing 6.5.

Following the procedure flow chart, Figure 6.1, the outputs of performing Steps 3-5 in `pyMMSFoam` are illustrated in Listings 6.3, 6.4, 6.5 and 6.6, respectively. The generated codes must

```
"(patch1|patch2|patch3)"
{
      type codedMixed;
      refValue uniform 0;
      refGradient uniform 0;
      valueFraction uniform 0;

      name T_Neumann;

      code
      #{
            const fvPatch& boundaryPatch = patch();
            const vectorField& Cf = boundaryPatch.Cf();
            const vectorField nf = patch().nf();
            const scalar t = this->db().time().value();

            forAll(this->patch(), faceI)
            {
                  const scalar x = Cf[faceI].x();
                  const scalar y = Cf[faceI].y();

                  const scalar tmp0 = 300*Foam::sin(0.10*t + x*x + y*y);
                  const scalar dT_dx = -tmp0*x;
                  const scalar dT_dy = -tmp0*y;
                  const scalar dT_dz = 0;

                  const vector gradT (dT_dx, dT_dy, dT_dz);

                  const scalar normalGradient = gradT & nf[faceI] ;
                  this->refGrad()[faceI] = normalGradient;
                  this->valueFraction()[faceI] = scalar(0);
            }
      #};
}
```

Listing 6.4: Neumann boundary condition generated for the 2D unsteady heat transfer equation (`laplacianFoam` solver).

be copied into the `fvOptions` file, the corresponding variable file in the "0" folder and the function section of the `controlDict` file, respectively.

After defining the manufactured solution and computing the source term, the appropriate boundary condition functions and the `coded functionObject` to monitor the computed solution error, a set of OpenFOAM® test cases must be prepared to determine the solver convergence order. For this purpose, successively finer meshes or time-steps should be used for each of those test cases, following the procedure described in Section 6.2.3. For each of the test cases, the output errors as well and the associated mesh characteristic sizes (or the number of cells) or time-steps should be collected. After computing the ratio between the characteristic sizes of two successively finer meshes

```
functions
{
   errorNorm_T
   {
      type coded;
      libs (utilityFunctionObjects);
      writeControl writeTime;
      name analyticalSolution_T;
      codeWrite
      #{
         const volScalarField& T_find = mesh().lookupObject<volScalarField>("T");
         const volVectorField& C = mesh().C();
         const scalarField& V = mesh().V();

         volScalarField MMS_diff_T
         (
            IOobject
            (
               "MMS_diff_T",
               mesh().time().timeName(),
               mesh(),
               IOobject::NO_READ,
               IOobject::AUTO_WRITE
            ),
            mesh(),
            dimensionedScalar ("MMS_diff_T_", dimless, 0.0)
         );

         const Time& time = mesh().time();
         const scalar t = time.value();

         forAll(MMS_diff_T, cellI)
         {
            const scalar x = C[cellI].x();
            const scalar y = C[cellI].y();
            const scalar solution = 225.0 + 150*Foam::cos(0.10*t + x*x + y*y);

            MMS_diff_T[cellI] = mag(solution - T_find[cellI]);
         }

         Info<< "L1 norm is: " << gSum(MMS_diff_T*V)/gSum(V) << endl;

         Info<< "L2 norm is: "
            << sqrt(gSum(MMS_diff_T*MMS_diff_T*V)/gSum(V))<< endl;

         Info<< "Linf norm is: " << gMax(MMS_diff_T) << endl;

         MMS_diff_T.write();
      #};
   }
}
```

Listing 6.5: `coded functionObject` generated for the 2D unsteady heat transfer equation (`laplacianFoam` solver).

(or time-steps) according to Equation (6.6), the solver convergence order can be determined as in Equations (6.7) to (6.9).

6.4 Case studies

In this section, some test case studies are presented to illustrate the application of the proposed semi-automatic approach in OpenFOAM® for the verification of the `laplacianFoam` and `simpleFoam` solvers. For all the cases, a two-dimensional unitary square domain $\Omega = [0, 1]^2$ m, on the $x - y$ plane, is considered, as shown in Figure 6.2.

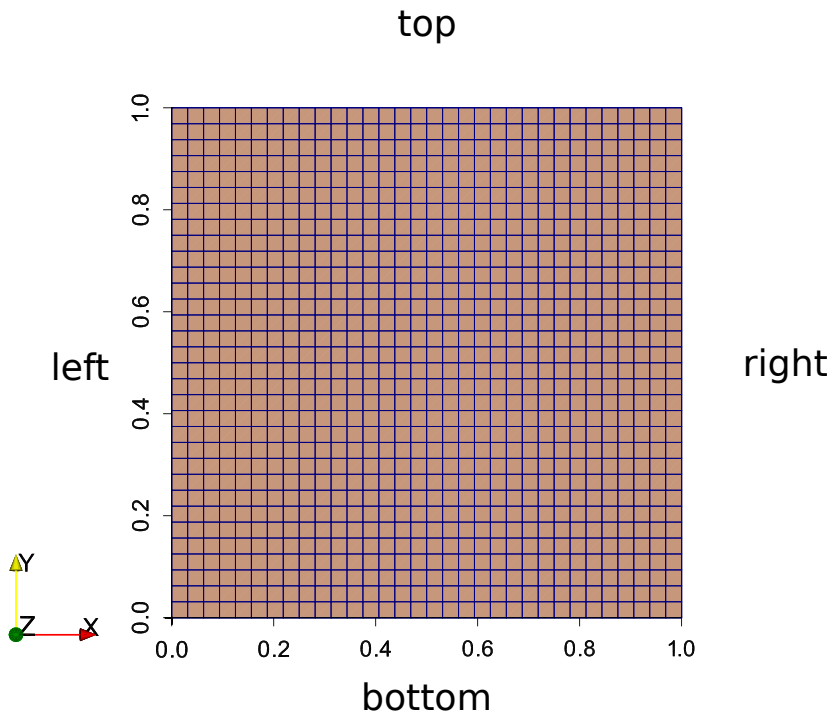


Figure 6.2: 2D domain with representation of the corresponding patches (dimensions in meters).

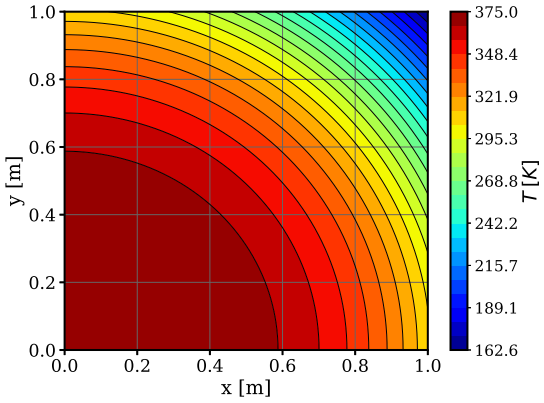
On the `top`, `bottom`, `left` and `right` patches of the domain, Dirichlet and/or Neumann boundary conditions are prescribed. On the patches normal to the z-axis, referred to as `front` and `back`, a boundary condition of type `empty` is prescribed to reduce the dimensionallity of the problem to 2D. In that regard, as is standard practice in OpenFOAM®, the generated meshes have a single cells layer along the z-direction.

The solver spatial convergence order is assessed with 5 grids, the coarser one having 32 elements along the x- and y-directions, and doubling the number of the cells in each direction for each successively finer grid.

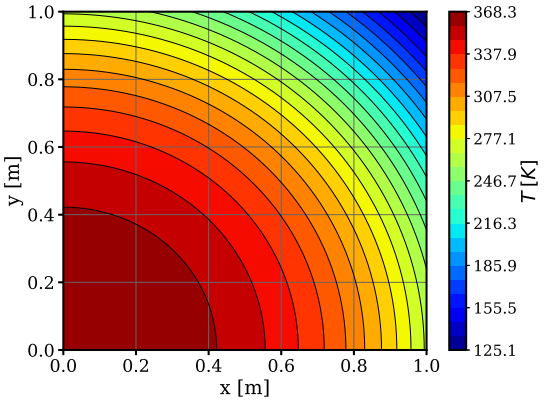
6.4.1 `laplacianFoam` test case studies

The `laplacianFoam` solver implements an unsteady/steady-state heat transfer model governed by Equation (6.1). Three test case studies are presented to verify the solver: a 2D steady-state test case and a 2D and a 1D unsteady test cases. For all, the manufactured solution for the temperature corresponds to Equation (6.1) and the associated source term is given in Equation (6.2). For the sake of simplicity only Dirichlet boundary conditions, Equation (6.3), are prescribed.

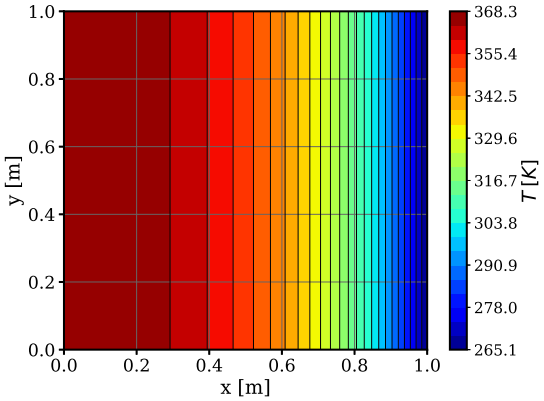
The `fvOptions` for the source term, the `codedFixedValue` for the boundary conditions, and the `coded functionObject` for the computation of the error norms, all generated with `pyMMSFoam`, are provided in Listings 6.2 to 6.5. Notice that $\omega = 0$ is set for the 2D steady-state test case study. Moreover, for the 1D unsteady test case study, only cells along the x-direction are generated and the y-coordinate dependence is removed from the manufactured solution, Equation (6.1). Finally, a thermal diffusivity coefficient of $D_T = 10^{-3}$ m^{2}s^{-1} is considered for all case studies. The resulting temperature field for the three test case studies is depicted in Figure 6.3.



(a) 2D steady-state test case study.



(b) 2D unsteady test case study at $t = 3$s.



(c) 1D unsteady test case study at $t = 3$s.

Figure 6.3: Manufactured solutions for the verification of the `laplacianFoam` solver.

For the three test case studies, the temporal and spatial discretization schemes employed for the

`laplacianFoam` solver are provided in Table 6.1. Moreover, the resulting system of linear equations is solved with a preconditioned conjugate gradient (PCG) method using a diagonal-based incomplete Cholesky (DIC) preconditioner. The associated parameters are given in Table 6.2.

Table 6.1: Temporal and spatial discretization schemes for the verification of the `laplacianFoam` solver (`fvSchemes` dictionary).

Parameter	Steady-state	Unsteady
Time derivative, $\partial\phi/\partial t$	`steadyState`	`Euler`
Gradient, $\nabla\phi$	`Gauss linear`	
Convective term, $\nabla \cdot (\phi\,\mathbf{u})$	`none`	
Laplacian, $\nabla \cdot (\Gamma\nabla\phi)$	`Gauss linear uncorrected`	
Interpolation from cell centers to faces	`linear`	
Surface-normal gradient, $\nabla_f^{\perp}\phi$	`none`	

Table 6.2: Solution methods for the verification of the `laplacianFoam` solver (`fvSolution` dictionary).

Parameter	
solver	`PCG`
preconditioner	`DIC`
tolerance	10^{-12}
relative tolerance	0.0

6.4.2 `simpleFoam` test case study

The governing equations solved in the `simpleFoam` solver correspond to the Navier-Stokes equations for a steady-state and incompressible flow [18], given as:

$$\nabla \cdot \mathbf{U} = 0, \tag{6.1}$$

$$\nabla \cdot (\mathbf{U} \otimes \mathbf{U}) - \nabla \cdot \mathbf{R} = -\nabla p + S_{\mathrm{U}}. \tag{6.2}$$

Equations (6.1) and (6.2) represent the mass conservation and the momentum balance, respectively, where for a 2D case:

- $\mathbf{U} \equiv (u, v) \equiv (u\,(x, y)\,, v\,(x, y))$ is the velocity vector,
- $\mathbf{R} \equiv \mathbf{R}\,(x, y)$ is the deviatoric stress tensor divided by the density,
- $p \equiv p\,(x, y)$ is the kinematic pressure (pressure divided by the density), and
- $S_U \equiv S_U\,(x, y)$ is the source term.

In the scope of the present study, the deviatoric stress tensor is defined as:

$$\mathbf{R} = \nu \left[\nabla \mathbf{U} + (\nabla \mathbf{U})^{\mathrm{T}} \right], \tag{6.3}$$

where ν is the kinematic viscosity (dynamic viscosity divided by the density).

The manufactured solution chosen for this case study is known in the literature as the Kovasznay Flow [20], given by:

$$\begin{aligned} u\left(x, y\right) &= 1 - \mathrm{e}^{\lambda x} \cos\left(2\pi y\right), \\ v\left(x, y\right) &= \frac{\lambda}{2\pi} \mathrm{e}^{\lambda x} \sin\left(2\pi y\right), \\ p\left(x, y\right) &= \frac{1}{2}\left(1 - \mathrm{e}^{2\lambda x}\right), \\ \lambda &= \frac{Re}{2} - \sqrt{\frac{Re^2}{4} + 4\pi^2}, \end{aligned} \tag{6.4}$$

where Re is the flow Reynolds number.

For this test case study, kinematic viscosity of $\nu = 0.01$ m^{2}s^{-1} is assumed. In this flow, the velocity satisfies the null divergence condition and, for $Re = 5$, the resulting velocity and kinematic pressure fields are depicted in Figure 6.4.

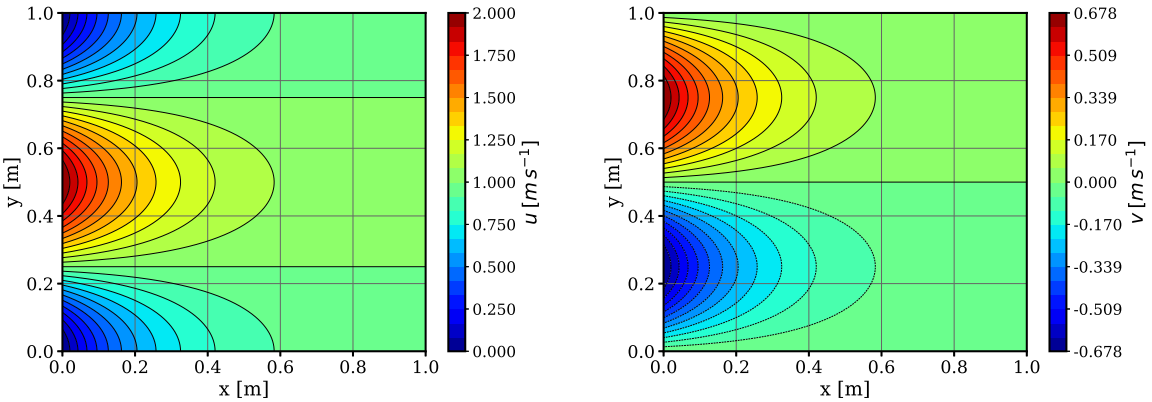
The code used to generate the source term, the boundary conditions and the `coded functionObjects` for the computation of the error norms, is shown in Listing 6.6.

From lines 6 to 11, the Reynolds number and the manufactured solution for the velocity components and pressure fields are defined. In lines 13 and 18, the velocity vector and stress tensor are calculated, respectively, and, in line 20, the momentum source term is computed. The advantages of using the `pyMMSFoam` package are clearly evidenced in such a convoluted manufactured solution, since it allows to avoid lengthy handmade and error-prone calculations. Finally, from lines 23 to 36, the `fvOptions` dictionary, boundary condition functions and `coded functionObject` are generated the OpenFOAM® test case study.

For this test case study, Dirichlet boundary conditions are prescribed for the velocity on the `left`, `top` and `bottom` patches of the domain, and for the pressure on the `right` patch. Moreover, Neumann boundary conditions are prescribed for the velocity on the `right` patch of the domain and for the pressure on `left`, `top` and `bottom` patches. The code required to setup these features in this case study are automatically generated by `pyMMSFoam`.

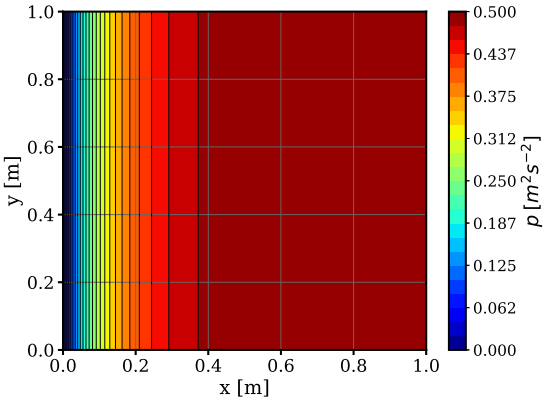
The temporal and spatial discretization schemes considered in this case study are presented in Table 6.3.

Is this test case the system of PDEs (6.1) and (6.2) is solved with the consistent version of the SIMPLE algorithm (SIMPLEC) [21] . For the resulting system of linear equations, a geometric algebraic multigrid method (GAMG) and a smooth solver were used to compute the approximate pressure and



(a) x-component of velocity.

(b) y-component of velocity.



(c) Kinematic pressure.

Figure 6.4: Manufactured solutions for the verification of the `simpleFoam` solver.

Table 6.3: Temporal and spatial discretization schemes for the verification of the `simpleFoam` solver (`fvSchemes` dictionary).

Parameter	Discretization scheme
Time derivative, $\partial\phi/\partial t$	`steadyState`
Gradient, $\nabla\phi$	`Gauss linear`
Convective term, $\nabla\cdot(\phi\,\mathbf{u})$	`Gauss linear`
Laplacian, $\nabla\cdot(\Gamma\nabla\phi)$	`Gauss linear uncorrected`
Interpolation from cell centers to faces	`linear`
Surface-normal gradient, $\nabla_f^{\perp}\phi$	`uncorrected`

velocity fields, respectively. Table 6.4 reports the solution methods considered and their respective parameters, and Table 6.5 provides the `SIMPLE` sub-dictionary parameters. Additionally, a relaxation factor of 0.9 was employed for the momentum balance equation, to assure appropriate calculation stability. For the solution stopping criterion, a tolerance of 10^{-9} for the initial residual of both the velocity and pressure is considered. Notice that such tolerance must be sufficiently small to guarantee that the obtained computed solution errors correspond only to discretization errors.

```
import sympy as sym
from sympy import sin, cos, exp, pi, sqrt
import pyMMSFoam as mms
from pyMMSFoam import x,y,z,t

Re = 5
Lambda = (Re/2) - sqrt( (Re**2/4) + 4*pi**2 )
u = 1 - exp(Lambda*x)*cos(2*pi*y)
v = (Lambda/(2*pi))*exp(Lambda*x)*sin(2*pi*y)
w = 0
p = 0.5*(1-exp(2*Lambda*x))

U = sym.Matrix([u,v,w])

# Momentum balance equation
nu = 0.01

R = nu*(mms.grad(U) + mms.grad(U).T)

S = mms.div(U*U.T) - mms.div(R) + mms.grad(p)

# Generate fvOptions
mms.generateFvOptions(S, "momentumSource", "U")

# Generate boundary conditions
# Velocity
mms.generateDirichletBoundaries(U, "U")
mms.generateNeumannBoundaries(U, "U")

# Pressure
mms.generateDirichletBoundaries(p, "p")
mms.generateNeumannBoundaries(p, "p")

# Generate functionObjects
mms.generateFunctionObject(U, "U")
mms.generateFunctionObject(p, "p")
```

Listing 6.6: `simpleFoam` MMS script.

Table 6.4: Solution methods for the verification of the `simpleFoam` solver (`fvSolution` dictionary).

Parameter	p	U
solver	GAMG	smoothSolver
smoother	DICGaussSeidel	symGaussSeidel
tolerance	10^{-11}	10^{-11}
relative tolerance	0.01	0.1

Table 6.5: Parameters for the SIMPLEC calculation procedure of the `simpleFoam` solver (`SIMPLE` sub-dictionary from `fvSolution` dictionary).

Parameter	Value
`momentumPredictor`	`yes`
`consistent`	`yes`
`residualControl`	`p` 10^{-9}
	`U` 10^{-9}

6.5 Results and Discussion

The results obtained from the test case studies presented in Section 6.4 to obtain the convergence order of the `laplacianFoam` and `simpleFoam` solvers are reported and discussed in the following subsections.

6.5.1 `laplacianFoam` test case studies

6.5.1.1 2D steady-state test case study

For this 2D steady-state case study, the errors in the L^1-, L^2- and L^∞-norms, obtained for successively finer meshes are reported in Table 6.6. The associated convergence orders between two consecutive finer meshes are given in Table 6.7. An overall second-order of spatial convergence is obtained for the `laplacianFoam` solver, regardless of the error norm. Moreover, these results are in accordance with the theoretical convergence orders for the spatial discretization schemes employed, which supports that the numerical code is correctly implemented for the meshes type employed.

Table 6.6: Error norms obtained in the 2D steady-state test case study for the `laplacianFoam` solver.

Mesh	N_C	h	E^1	E^2	E^∞
`Mesh 1`	1,024	3.13×10^{-2}	3.33×10^{-2}	3.84×10^{-2}	6.66×10^{-2}
`Mesh 2`	4,096	1.56×10^{-2}	8.33×10^{-3}	9.59×10^{-3}	1.70×10^{-2}
`Mesh 3`	16,384	7.81×10^{-3}	2.08×10^{-3}	2.40×10^{-3}	4.31×10^{-3}
`Mesh 4`	65,536	3.91×10^{-3}	5.21×10^{-4}	6.00×10^{-4}	1.09×10^{-3}
`Mesh 5`	262,144	1.95×10^{-3}	1.30×10^{-4}	1.50×10^{-4}	2.73×10^{-4}

6.5.1.2 2D unsteady test case study

For the 2D unsteady test case study, the computed solution error increases at every time-step due to the discretization in time, but the same theoretical convergence orders are expected regardless of the last simulated time. Therefore, for the solver verification and to assess the convergence order

Table 6.7: Convergence orders obtained in the 2D steady-state test case study for the `laplacianFoam` solver.

Mesh	O^1	O^2	O^∞
`Mesh 1` $\longrightarrow$ `Mesh 2`	2.00	2.00	1.97
`Mesh 2` $\longrightarrow$ `Mesh 3`	2.00	2.00	1.98
`Mesh 3` $\longrightarrow$ `Mesh 4`	2.00	2.00	1.99
`Mesh 4` $\longrightarrow$ `Mesh 5`	2.00	2.00	1.99

under mesh refinement, an appropriate time-step size should be determined to avoid the computed solution error stagnating due to the temporal discretization error, as described in Section 6.2. Moreover, notice that the Euler discretization scheme [22] employed to discretize the time derivative is first-order accurate, whereas the spatial discretization schemes considered have a theoretical second-order of accuracy. Therefore, without a careful choice of the time-step size, the verification of the spatial convergence order in this unsteady test case might be inappropriate. Accordingly, for this test case, the value of the time-step size was iteratively decreased until observing that the computed solution error (considering only the reported significant digits) remain unchanged for all the employed meshes. In that regard, a time-step size of $\Delta t = 10^{-7}$ s was chosen and a final time of $t^{\mathrm{F}} = 3$ s was considered for the simulations.

The errors in the L^1-, L^2- and L^∞-norms obtained for successively finer meshes, just for the last calculated instant of time, are reported in Table 6.8, while the associated convergence orders between two consecutive finer meshes are given in Table 6.9. As observed, the `laplacianFoam` solver provides an overall second-order spatial convergence, regardless of the error norm. As in the previous case study, these results are in accordance with the theoretical convergence orders for the considered spatial discretization schemes. Moreover, the procedure employed for the choice of the time-step size was shown to be effective for the correct assessment of the spatial convergence order of the `laplacianFoam` solver.

Table 6.8: Error norms obtained in the 2D unsteady test case study for the `laplacianFoam` solver.

Mesh	N_{C}	h	E^1	E^2	E^∞
`Mesh 1`	1,024	3.13×10^{-2}	6.98×10^{-3}	1.26×10^{-2}	4.90×10^{-2}
`Mesh 2`	4,096	1.56×10^{-2}	1.76×10^{-3}	3.18×10^{-3}	1.30×10^{-2}
`Mesh 3`	16,384	7.81×10^{-3}	4.42×10^{-4}	7.97×10^{-4}	3.34×10^{-3}
`Mesh 4`	65,536	3.91×10^{-3}	1.11×10^{-4}	1.99×10^{-4}	8.44×10^{-4}
`Mesh 5`	262,144	1.95×10^{-3}	3.09×10^{-5}	5.04×10^{-5}	2.12×10^{-4}

6.5.1.3 1D unsteady test case study

Contrarily to the previous, successively smaller time-step sizes, and a fixed mesh characteristic size, are considered in this test case study, to assess the temporal convergence order of the

Table 6.9: Convergence orders obtained in the 2D unsteady test case study for the `laplacianFoam` solver.

Mesh	O^1	O^2	O^∞
`Mesh 1` $\rightarrow$ `Mesh 2`	1.98	1.99	1.91
`Mesh 2` $\rightarrow$ `Mesh 3`	2.00	2.00	1.96
`Mesh 3` $\rightarrow$ `Mesh 4`	2.00	2.00	1.98
`Mesh 4` $\rightarrow$ `Mesh 5`	1.84	1.98	1.99

`laplacianFoam` solver. For that purpose, an appropriate mesh characteristic size was iteratively determined following the procedure described in Section 6.2, to avoid the computed solution error stagnating due to the spatial discretization error. The chosen mesh consists of $8,192$ cells along the x-direction.

The errors in the L^1-, L^2- and L^∞-norms obtained for successively smaller time-step sizes, for the last calculated instant of time, are reported in Table 6.10, while the associated convergence orders between two consecutive smaller time-step sizes are given in Table 6.11. As expected, the first-order of convergence is generally achieved, for all error norms. This emphasizes the simplicity and practicality of the proposed semi-automatic approach to assess the convergence order of a solver in OpenFOAM® and to detect possible numerical issues or implementation inconsistencies.

Table 6.10: Errors obtained in the 1D unsteady test case study for the `laplacianFoam` solver.

N_C	Δt [s]	E^1	E^2	E^∞
8,192	5×10^{-3}	8.82×10^{-3}	9.17×10^{-3}	1.10×10^{-2}
	5×10^{-4}	8.82×10^{-4}	9.17×10^{-4}	1.10×10^{-3}
	5×10^{-5}	8.81×10^{-5}	9.17×10^{-5}	1.10×10^{-4}
	5×10^{-6}	8.76×10^{-6}	9.15×10^{-6}	1.10×10^{-5}

Table 6.11: Convergence orders obtained in the 1D unsteady test case study for the `laplacianFoam` solver.

Δt [s]	O^1	O^2	O^∞
$5 \times 10^{-3} \rightarrow 5 \times 10^{-4}$	1.00	1.00	1.00
$5 \times 10^{-4} \rightarrow 5 \times 10^{-5}$	1.00	1.00	1.00
$5 \times 10^{-5} \rightarrow 5 \times 10^{-6}$	1.00	1.00	1.00

6.5.2 `simpleFoam` test case study

For the sake of simplicity, only the errors in the L^2-norm are presented in this test case study, to verify the convergence order of the `simpleFoam` solver for both velocity vector components and pressure fields. In that regard, the errors in the L^2-norm obtained for successively finer meshes are reported in Table 6.12, while the associated convergence orders between two consecutive finer

meshes are given in Table 6.13. For the velocity, the second-order of convergence is achieved for both components, although more consistently for v, whereas for u the convergence order is slightly below with coarser grids. For the pressure, the second-order of convergence is obtained but it seems to slightly deteriorate as finer grids are considered. Indeed, due to the pressure-velocity coupling in the incompressible Navier-Stokes equations, discretization techniques might provide convergence orders below the theoretical values, which is in accordance with the information provided in the literature [4, 23, 24]. Therefore, it is important to identify the theoretically expected convergence orders, not only according to the employed discretization schemes but also considering the equations being solved. However, such a detailed analysis is out of the scope of the present work. Nevertheless, it can be easily analyzed with the support of the proposed semi-automatic approach.

Table 6.12: Error norms obtained for u, v, and p in the 2D steady-state test case study for the `simpleFoam` solver.

Mesh	N_C	h	u	v	p
`Mesh 1`	1,024	3.13×10^{-2}	6.04×10^{-2}	5.43×10^{-3}	2.29×10^{-2}
`Mesh 2`	4,096	1.56×10^{-2}	1.68×10^{-2}	1.34×10^{-3}	5.81×10^{-3}
`Mesh 3`	16,384	7.81×10^{-3}	4.48×10^{-3}	3.37×10^{-4}	1.55×10^{-3}
`Mesh 4`	65,536	3.91×10^{-3}	1.16×10^{-3}	8.43×10^{-5}	4.38×10^{-4}
`Mesh 5`	262,144	1.95×10^{-3}	2.97×10^{-4}	2.11×10^{-5}	1.31×10^{-4}

Table 6.13: Convergence orders obtained for u, v, and p in the 2D steady-state test case study for the `simpleFoam` solver.

Mesh	u	v	p
`Mesh 1` $\longrightarrow$ `Mesh 2`	1.85	2.01	1.98
`Mesh 2` $\longrightarrow$ `Mesh 3`	1.90	2.00	1.90
`Mesh 3` $\longrightarrow$ `Mesh 4`	1.95	2.00	1.82
`Mesh 4` $\longrightarrow$ `Mesh 5`	1.97	2.00	1.74

6.6 Conclusions

The verification of a numerical method and the associated implementation is an essential step in any solver development, to ensure that the implemented code is correctly solving the prescribed mathematical model. In that regard, assessing that the computed solution error converges with some order to the underlying exact solution, as the mesh characteristic size and/or time-step size decreases, is a necessary condition. In the general case, finding analytic solutions for such a verification process can be a cumbersome task, especially for complex geometries and/or mathematical models. For the more demanding cases, the method of manufactured solutions provides a practical, simple and versatile framework for the verification procedure of numerical codes

(discretization schemes, solution methods, boundary conditions, material models, etc.). In the present work, a semi-automatic approach was proposed based on the method of manufactured solutions for the verification of solvers in OpenFOAM®.

For this aim, a `Python` package, `pyMMSFoam`, was developed to support the calculation of source terms, boundary conditions and functions to calculate the errors associated with the numerical method implemented to solve a prescribed mathematical model. Moreover, `pyMMSFoam` automatically generates the associated C-code and case setup `dictionaries`, as well as `coded functionObjects` to compute the errors in different norms, allowing a less invasive, less error-prone and more flexible methodology for code verification. The application of the proposed approach was illustrated for the verification of the `laplacianFoam` and `simpleFoam` solvers of OpenFOAM®, with different test case studies, consisting of 1D and 2D, steady-state and unsteady problems, employing structured meshes. The approach proved to be very practical to apply and provides simple and versatile means of assessing the convergence orders of a numerical code. It is important to notice that consecutive lower residual tolerances for the solution method should be used in the calculations until confirming that the computed solution is influenced only by the discretization schemes and not by the algorithm that solves the system of linear equations.

Finally, `pyMMSFoam` is open to the community and the authors welcome suggestions and improvements.

Appendix

Availability of data and material

The data that support the findings of this study are openly available at `https://github.com/Computational-Rheology/MMS_Generator`.

References

[1] Patrick J Roache. *Verification and validation in computational science and engineering*, volume 895. Hermosa Albuquerque, NM, 1998.

[2] William L Oberkampf and Christopher J Roy. *Verification and validation in scientific computing*. Cambridge University Press, 2010.

[3] Verification and validation. `https://www.openfoam.com/documentation/guides/latest/doc/guide-verification-validation.html`. Accessed: 2021-08-20.

[4] Kambiz Salari and Patrick Knupp. *Code Verification by the Method of Manufactured Solutions*. Sandia National Laboratories, Albuquerque, NM 87 185-0825, 2000.

[5] Patrick J. Roache. Code verification by the method of manufactured solutions. *Journal of Fluids Engineering, Transactions of the ASME*, 124(1):4–10, 2002.

[6] C. J. Roy, C. C. Nelson, T. M. Smith, and C. C. Ober. Verification of Euler/Navier-Stokes codes using the method of manufactured solutions. *International Journal for Numerical Methods in Fluids*, 44(6):599–620, 2004.

[7] Bruno Blais and François Bertrand. On the use of the method of manufactured solutions for the verification of CFD codes for the volume-averaged Navier-Stokes equations. *Computers and Fluids*, 114:121–129, 2015.

[8] H. Noriega, F. Guibault, M. Reggio, and R. Magnan. A case-study in open-source CFD code verification, Part I: Convergence rate loss diagnosis. *Mathematics and Computers in Simulation*, 147:152–171, 2018.

[9] H. Noriega, F. Guibault, M. Reggio, and R. Magnan. A case-study in open-source CFD code verification. Part II: Boundary condition non-orthogonal correction. *Mathematics and Computers in Simulation*, 147:172–193, 2018.

[10] Aniruddha Choudhary, Christopher J. Roy, Edward A. Luke, and Subrahmanya P. Veluri. Code verification of boundary conditions for compressible and incompressible computational fluid dynamics codes. *Computers & Fluids*, 126:153–169, mar 2016.

[11] Rupert Fisch, Jörg Franke, Roland Wüchner, and Kai-Uwe Bletzinger. Code verification examples of a fully geometrical nonlinear membrane element using the method of manufactured solutions. In *Textile Composites and Inflatable Structures VI - Proceedings of the 6th International Conference on Textile Composites and Inflatable Structures, Structures Membranes*, pages 102–113, 2013.

[12] Rupert Fisch, Jörg Franke, Roland Wüchner, and K. Bletzinger. Code verification of a partitioned FSI environment for wind engineering applications using the method of manufactured solutions. *11th World Congress on Computational Mechanics, WCCM 2014, 5th European Conference on Computational Mechanics, ECCM 2014 and 6th European Conference on Computational Fluid Dynamics, ECFD 2014*, (Wccm Xi):2186–2197, 2014.

[13] Florian Habla, C. Fernandes, Maximilian Maier, Lennart Densky, L.L. Ferrás, Ananth Rajkumar, O.S. Carneiro, Olaf Hinrichsen, and J. Miguel Nóbrega. Development and validation of a model for the temperature distribution in the extrusion calibration stage. *Applied Thermal Engineering*, 100:538–552, 2016.

[14] Stefan Radman, Carlo Fiorina, and Andreas Pautz. Development of a novel two-phase flow solver for nuclear reactor analysis: algorithms, verification and implementation in OpenFOAM. *Nuclear Engineering and Design*, 379:111178, 2021.

[15] Malik Hassanaly, Heeseok Koo, Christopher F. Lietz, Shao Teng Chong, and Venkat Raman. A minimally-dissipative low-Mach number solver for complex reacting flows in OpenFOAM. *Computers and Fluids*, 162:11–25, 2018.

[16] *SymPy*. `https://www.sympy.org/en/index.html`. Accessed: 2021-08-20.

[17] *laplacianFoam*. `https://www.openfoam.com/documentation/guides/latest/doc/guide-applications-solvers-basic-laplacianFoam.html`. Accessed: 2021-08-20.

[18] *simpleFoam*. `https://www.openfoam.com/documentation/guides/latest/doc/guide-applications-solvers-incompressible-simpleFoam.html`. Accessed: 2021-08-20.

[19] Sympy's common subexpression elimination. `https://docs.sympy.org/latest/modules/simplify/simplify.html?highlight=cse#module-sympy.simplify.cse_main`. Accessed: 2021-08-20.

[20] L. I. G. Kovasznay. Laminar flow behind a two-dimensional grid. *Mathematical Proceedings of the Cambridge Philosophical Society*, 44(1):58–62, 1948.

[21] *Solution and algorithm control*. `https://www.openfoam.com/documentation/user-guide/6-solving/6.3-solution-and-algorithm-control`. Accessed: 2021-08-20.

[22] Euler scheme. `https://www.openfoam.com/documentation/guides/latest/doc/guide-schemes-time-euler.html`. Accessed: 2021-08-20.

[23] Ricardo Costa, Stéphane Clain, Gaspar J Machado, and Raphaël Loubère. A very high-order accurate staggered finite volume scheme for the stationary incompressible Navier–Stokes and Euler equations on unstructured meshes. *Journal of Scientific Computing*, 71(3):1375–1411, 2017.

[24] Ricardo Costa, Stéphane Clain, and Gaspar J. Machado. A sixth-order finite volume scheme for the steady-state incompressible Stokes equations on staggered unstructured meshes. *Journal of Computational Physics*, 349:501–527, 2017.

Chapter 7 An Open-Source Framework for Modeling the Evolution of Fiber Orientation

Abstract:

Modeling fiber orientation plays a crucial role in predicting the behavior of fiber reinforced thermoplastic materials. The equation that governs the evolution of the fiber orientation is hyperbolic in nature and requires handling of fourth-order tensors, which are currently unavailable in OpenFOAM®. The current work explores the possibility of using OpenFOAM® and open-source symbolic computation for modeling the evolution of fiber orientation. For this purpose, a `functionObject` was programmed to work as a plug-in for any OpenFOAM® incompressible flow solver and compute the evolution of fiber orientation in a decoupled manner. Several fiber orientation models and closure relations available in the literature were implemented in the tool, which were verified by comparing their predictions with independent results for a single material point obtained by numerically integrating the associated governing equations.

Keywords: Fiber orientation modeling; OpenFOAM® plugin solver;

This chapter was adapted from Ramoa, B., Costa, R., Chinesta, F., & Nóbrega, J. M. (2025). An Open-Source Framework for Modeling the Evolution of Fiber Orientation. OpenFOAM® Journal, 5, 17-37. https://doi.org/10.51560/ofj.v5.131

7.1 Introduction

Thermoplastic materials are tied to our daily lives, encompassing a wide range of applications, from household products to advanced technical appliances. To meet the permanent demand for enhanced products, fiber-reinforced thermoplastic materials are often employed to achieve specific properties (property weighted by the density) that cannot be obtained with conventional materials. Industries such as automotive heavily rely on this technology to achieve lightweight designs, essential for optimizing fuel efficiency [1].

The final properties of thermoplastic parts depend on the microstructure developed during processing. For FRTM, the effects of fiber length, concentration, and orientation play a significant role in the performance of the final part. Particularly, fiber orientation has been extensively studied throughout the last decades [2–6], leading to the development of several phenomenological models [7–13] based on the seminal work of Jeffery [14] to quantify the state of orientation in FRTM parts. During the product development phase, knowledge of this final state of orientation is crucial for structural analysis and design, as it directly affects the parts' mechanical performance [15–17].

In the context of OpenFOAM® only a few works have been dedicated to fiber orientation modeling, where only the Folgar-Tucker (FT) [7] and reduce strain closure [8] models in tensor-based formulations have been reported [18–21]. Given the open-source character and outreach of OpenFOAM®, this work presents a novel strategy for modeling the fiber orientation evolution in incompressible fluid flows using the OpenFOAM® framework. The developed plug-in can be utilized for injection molding, compression molding, and any other solver that involves fluid flow. The objective is to incorporate the capability of predicting fiber orientation as an extension, without making modifications to an existing solver. The approach involves creating a `functionObject` to solve the three-dimensional orientation evolution equation with state-of-the-art models and closure approximations in a decoupled manner, using a one-way interaction, where the velocity field affects the fiber orientation evolution but the computed orientation will not influence the relation between the velocity gradient and the stress tensor ($\boldsymbol{\tau}$), i.e., the constitutive model, required to calculate the velocity field. Nevertheless, the strategy employed is flexible enough for the interested user to extend the developed classes to account for the effect of fiber orientation into the stress tensor calculation [22]. An initial version of this methodology was first documented in [23], a master's thesis carried out under the author's co-supervision. This work presents a full re-implementation, extends the range of available models, and integrates the methodology into a plug-in solver.

The remaining paper is organized as follows: In Sections 7.2 and 7.3, the proposed approach to implement fiber orientation modeling within the OpenFOAM® framework is presented. Sections 7.4 and 7.5 showcase and discuss case studies involving simple-shear and the flow in a center-gated disk to validate the numerical implementations. Finally, Section 7.6 summarizes the main conclusions drawn from this work.

7.2 Modeling fiber orientation within the OpenFOAM ® framework

Tensor-based fiber orientation modeling is a pure advective problem, hence the formulations presented in Chapter 2 are better suited for a Lagrangian approach [24]. To calculate the fiber orientation evolution equation in an Eulerian framework, the definition of the material derivative is used:

$$\dot{\mathbf{A}} = \frac{\mathrm{D}\mathbf{A}}{\mathrm{D}t} = \frac{\partial \mathbf{A}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{A} = \frac{\partial \mathbf{A}}{\partial t} + \nabla \cdot (\mathbf{v}\,\mathbf{A}) - (\nabla \cdot \mathbf{v})\,\mathbf{A}. \tag{7.1}$$

Due to the incompressibility assumption, $(\nabla \cdot \mathbf{v})\,\mathbf{A}$ is zero. The equation that governs the evolution of $\mathbf{A}$ can be seen as a component-wise advection equation, coupled by a source term given by the right-hand side of the fiber orientation model. In order to introduce fiber orientation modeling into OpenFOAM®, some considerations have to be made. In Section 2, the velocity gradient was defined as $L_{ij} = \frac{\partial v_i}{\partial x_j}$, however in OpenFOAM®, $\nabla v_{ij} = L_{ji} = \frac{\partial v_j}{\partial x_i}$, which requires the adaptation of the the vorticity tensor, $\mathbf{W}$, whose definition in the OpenFOAM® framework is the anti-symmetric of the one defined in Section 2.

Currently, OpenFOAM® does not provide a fourth-order tensor class, and only the `extend` branch offers a simplified fourth-order tensor class, with nine independent components and a limited set of operations. However, to be able to calculate the evolution of fiber orientation, it is necessary to know the contraction $D_{kl}A_{ijkl}$. Therefore, A_{ijkl} must be computed based on a closure relation, as described in Section 2.2. One potential solution for addressing this issue is to implement a comprehensive class to handle the operations. Alternatively, recognizing that $D_{kl}A_{ijkl}$ is a second-order tensor, it can be introduced as a source term in the evolution equation, and symbolic computation can be used to infer the required operations and generate the necessary code. In this work the latter approach was adopted.

The framework for fiber orientation modeling was developed considering two abstract classes, `fiberOrientationModel` and `closureModel`. These were developed following the structure of the viscosity models available in `$FOAM_SRC/transportModels/incompressible/viscosityModels`, with run-time selection. For each class the required minimum attributes and member functions were added to model fiber orientation. Accordingly, the `fiberOrientationModel` class contains a dictionary for collecting input data from the user, fields that are common to all models ($\mathbf{A}$, $\mathbf{L}$, $\mathbf{D}$, $\mathbf{W}$, $\dot{\gamma}$, and $\mathbb{A} : \mathbf{D}$), a pure virtual `solve` method for assembling and solving the system of linear equations and some other minor methods to assure its functionality. The `closureModel` class includes one pure virtual function named `computeClosure` to compute the product $D_{kl}A_{ijkl}$ and one virtual function named `computeRSCClosure` to compute $\left[A_{ijkl} + (1-\kappa)\left(L_{ijkl} - M_{ijmn}A_{mnkl}\right)\right] D_{kl}$, required by the RSC model, together with additional minor functionalities.

Based on the above structure, each fiber orientation model that derives from the

`fiberOrientationModel` class should: (i) specify a sub-dictionary to collect model-specific information and any required additional fields (e.g. $\mathbf{C}$), and (ii) implement the corresponding `solve` method. Additionally, each closure model (hybrid, ORE, IBOF, etc.), which derives from the `closureModel` class, should implement the `computeClosure` method. If one decides to make the closure also available for the RSC approach, the method `computeRSCClosure` should also be defined.

The framework implemented in OpenFOAM® is depicted in Figure 7.1. Currently, these classes are wrapped in a `functionObject` (FO) to work as a plug-in for any incompressible flow solver. At the end of each specified time step, upon calling the `execute` method from the FO, the `solve` function from a specific `fiberOrientationModel` will be executed, assembling and solving the corresponding system of linear equations. From the user perspective, within the `functions` section in the `controlDict` dictionary, in addition to the usual FO parameters, it is required to define a `dictionary` named `fiberOrientationProperties` that indicates the fiber model (parameter `model`), fiber parameters (`subdictionary <fiberModel>Properties`), and closure model (parameter `closureModel`).

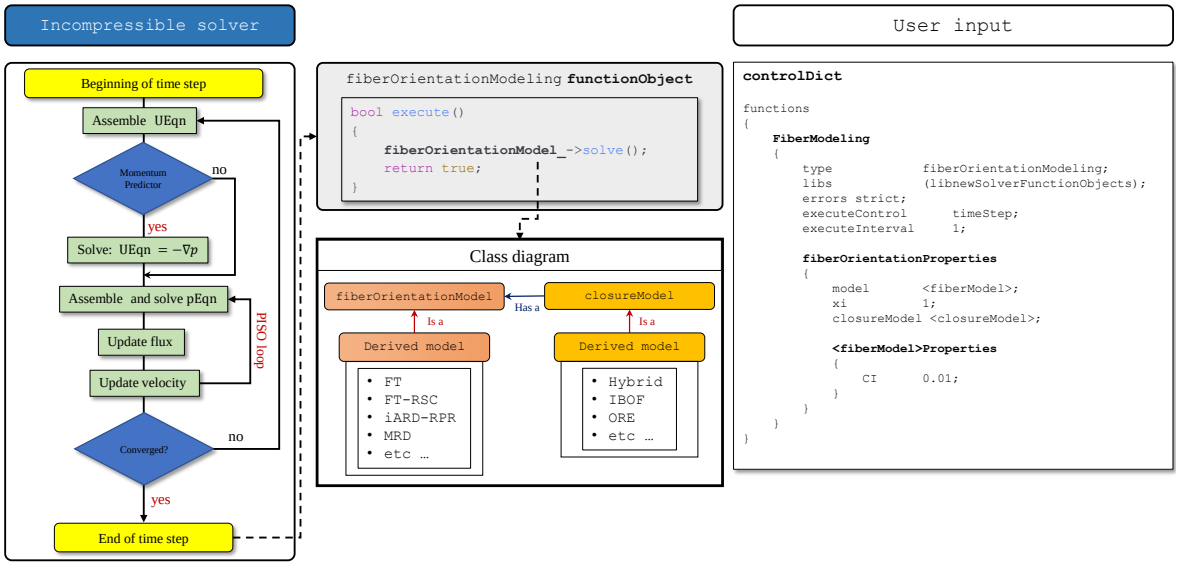


Figure 7.1: Concept for the `functionObject` for modeling fiber orientation.

Following the OpenFOAM® modeling approach, the final equation syntax is similar to the mathematical notation. As an example, the final form of the FT model [7], which is included in `solve()` method, is given in Listing 7.1. The variable `alphaClip` is used a binary mask that activates the transport equation if $\alpha \geq \alpha_{\text{threshold}}$. If no `phase` name is provided it will default to 1 everywhere. Finally, `D_doubleDot_A4_` contains the calculation of $D_{kl}A_{ijkl}$, whose calculation is described in Section 7.3.

The resultant framework for the implemented fiber orientation modeling should be easily extendable to other fiber orientation and closure models. Currently, due to the validity of the implemented formulations, the available closure models are only valid for 3D cases.

```
fvSymmTensorMatrix dA2dtEqn
(
   fvm::ddt(A2_) + fvm::div(phi_, A2_, divScheme)
   ==
   alphaClip*(
             symm(
                    (W_ & A2_) - (A2_ & W_)
                   +xi_*(
                          (D_ & A2_) + (A2_ & D_) - (2.0*D_doubleDot_A4_)
                      )
                )
             + 2.0*CI_*shrRate_*(symmTensor::I - 3.0*A2_)
         )
);
```

Listing 7.1: Folgar-Tucker fiber orientation model.

7.3 Closure model with symbolic computation

The open-source library SymPy v1.11 [25] was used to perform the operations that OpenFOAM® does not have, namely the contractions between fourth- and second-order tensors. This rich-featured library for symbolic computation is written in Python language and provides functionalities to convert the symbolic code into C-style code, which can be used in OpenFOAM® [26].

In the developed Python scripts provided in the repository folder `fiberClosureSymbolic`, a vector is treated as a $[3 \times 1]$ matrix, a second-order tensor as a $[3 \times 3]$ matrix, and a fourth-order tensor as a dense multidimensional array with shape $[3 \times 3 \times 3 \times 3]$.

The Python script required to generate the C-code that calculates the closure method (`computeClosure`), for the hybrid closure model in the `hybrid` class, is illustrated in Listing 7.2. In Lines 1–3 of Listing 7.2, the SymPy module is imported with the alias `sym` and the functions `symm`, `fourthOrderIndexPermutation` and `generateCCode` from the module `fiberOrientationModelling_Tools`. These functions are responsible for generating a symmetric second-order tensor, performing index permutations within the fourth-order tensor and generating C-code, respectively.

Listing 7.2 comprises two functions, `computeHybridClosure` and `main`. The former computes the hybrid closure symbolically, while the latter, creates the symbolic variables, performs the tensor contraction and generates the associated C-code.

In the `computeHybridClosure` function, a matrix symbol is created representing a second-order tensor (Line 7), symmetry conditions are applied (Line 9), and a second-order identity tensor is defined (Line 11). Finally, in Lines 13–36, the hybrid closure is computed following Equation (2.16), to obtain A_{ijkl}^{hyb}.

```
1  import sympy as sym
2
3  from fiberOrientationModelling_Tools import symm, fourthOrderIndexPermutation, generateCCode
4
5  def computeHybridClosure():
6
7     a = sym.MatrixSymbol('A',3,3)
8
9     A2 = symm(sym.Matrix(a))
10
11    I = sym.eye(3)
12
13    A4_linear = (
14              sym.Rational(-1, 35)*
15              (
16                  fourthOrderIndexPermutation('ijkl', sym.tensorproduct(I,I))
17                + fourthOrderIndexPermutation('ikjl', sym.tensorproduct(I,I))
18                + fourthOrderIndexPermutation('iljk', sym.tensorproduct(I,I))
19              )
20
21              + sym.Rational(1, 7)*
22              (
23                  fourthOrderIndexPermutation('ijkl', sym.tensorproduct(A2,I))
24                + fourthOrderIndexPermutation('ikjl', sym.tensorproduct(A2,I))
25                + fourthOrderIndexPermutation('iljk', sym.tensorproduct(A2,I))
26                + fourthOrderIndexPermutation('klij', sym.tensorproduct(A2,I))
27                + fourthOrderIndexPermutation('jlik', sym.tensorproduct(A2,I))
28                + fourthOrderIndexPermutation('jkil', sym.tensorproduct(A2,I))
29              )
30            )
31
32    A4_quadratic = sym.tensorproduct(A2, A2)
33
34    f = 1.0 - 27*A2.det()
35
36    A4_hybrid = f*A4_quadratic + (1.0 - f)*A4_linear
37
38    return A4_hybrid
39
40 def main():
41    d = sym.MatrixSymbol('D',3,3)
42
43    D = symm(sym.Matrix(d))
44
45    A4 = computeHybridClosure()
46
47    D_doubleDot_A4 = sym.tensorcontraction(sym.tensorproduct(D, A4),(0,4),(1,5))
48
49    generateCCode(sym.Matrix(D_doubleDot_A4))
50
51 if __name__ == "__main__":
52    main()
```

Listing 7.2: Symbolic computation of the hybrid closure model.

In the `main` function, in Lines 41–43, a matrix representing the second-order tensor $\mathbf{D}$ is created. In Line 45 the `computeHybridClosure` function is executed, returning the computation of the hybrid closure, and, in Line 47, the contraction $D_{kl}A_{ijkl}$ is performed. To conclude this function, in Line 49, C-code is generated from the symbolic computations.

The derivation of the C-code to implement the `computeRSCClosure` method in the `Hybrid` class is illustrated in Listings 7.3 and 7.4. To compute the RSC closure, a script with a general framework was developed through the `RSC` function. This method takes a closure function as an input argument and performs the required symbolic computations. In Lines 2–8 of Listing 7.3, the

slow-down symbolic variable, κ, is created as well as variables that will hold the eigenvalues and eigenvectors of $\mathbf{A}$. Although not required by the RSC approach, some orientation models and associated closures require the eigenvalues to be sorted in descending order and to have the eigenvectors as column vectors. Within the OpenFOAM® framework, the eigenvalues of a symmetric second-order tensor are computed and sorted in ascending order and the corresponding eigenvectors stored in a row-wise manner. The order of computations within OpenFOAM® was taken into account in the symbolic computations (see Lines 10–21).

In Line 23 the closure relation (the argument of the RSC function) is executed to obtain A_{ijkl}. Place holders for tensors $\mathbb{M}$ (Line 25) and $\mathbb{L}$ (Line 26) are created and the respective expressions are formulated (Lines 28–35) to get the expression between square brackets of the RSC model, Equation (2.14a).

```
1  def RSC(closureFunc):
2     k = sym.Symbol('k')
3
4     e = sym.MatrixSymbol('eigenVector',3,3)
5     eigVec = sym.Matrix(e)
6
7     w = sym.MatrixSymbol('eigenValue',3,1)
8     eigVal = sym.Matrix(w)
9
10    # Invert order to be consistent with OpenFOAM
11    tmpEigVal = eigVal.copy()
12
13    # sort eigenValues
14    eigVal[0] = tmpEigVal[2]
15    eigVal[2] = tmpEigVal[0]
16
17    # sort eigenVectors
18    eigVec = eigVec.T
19    tmpEigVec = eigVec.copy()
20    eigVec[:, 0] = tmpEigVec[:, 2]
21    eigVec[:, 2] = tmpEigVec[:, 0]
22
23    A4 = closureFunc()
24
25    M = sym.MutableDenseNDimArray.zeros(3,3,3,3)
26    L = sym.MutableDenseNDimArray.zeros(3,3,3,3)
27
28    for i in range(3):
29       e_vector = sym.Array(sym.flatten(eigVec[:, i]))
30       M += sym.tensorproduct(e_vector, e_vector, e_vector, e_vector)
31       L += eigVal[i]*sym.tensorproduct(e_vector, e_vector, e_vector, e_vector)
32
33    tmp1 = sym.tensorcontraction( sym.tensorproduct(M, A4), (2,4), (3,5) )
34
35    tensorCombination = A4 + (1-k)*(L- tmp1)
36
37    return tensorCombination
```

Listing 7.3: Symbolic computation of the RSC closure model.

The hybrid closure calculation used in the `computeRSCClosure` method is shown in Listing 7.4, which comprises only the `main` function. Similarly to the one shown in Listing 7.2, upon creating A_{ij} (Lines 7–9), and $[\mathbb{A} + (1 - \kappa)\,(\mathbb{L} - \mathbb{M} : \mathbb{A})]$ (Line 11), following the RSC framework, the contraction with the rate-of-deformation tensor is performed (Line 13). Finally, in Line 15, the associated C-code is generated.

```
1  import sympy as sym
2  from fiberOrientationModelling_symbolicComputationTools import symm,
       ↪ generateCCode
3  from hybricClosure import computeHybridClosure
4  from rsc import RSC
5
6  def main():
7      d = sym.MatrixSymbol('D',3,3)
8
9      D = symm( sym.Matrix(d) )
10
11     A4 = RSC(computeHybridClosure)
12
13     D_doubleDot_A4 = sym.tensorcontraction(sym.tensorproduct(D,
           ↪ A4),(0,4),(1,5))
14
15     generateCCode(sym.Matrix(D_doubleDot_A4))
16
17 if __name__ == "__main__":
18     main()
```

Listing 7.4: Symbolic computation of the hybrid closure model with RSC.

7.4 Case studies

This section presents the analysis performed to verify the implementation of the fiber orientation plug-in, implemented for modeling the evolution of fiber orientation, which was undertaken with two case studies: simple-shear flow and the flow in a center-gated disk. For these, an adaptation of the solver `pimpleFoam` was used. In both cases, because an analytic solution for the velocity field is available, the `pimpleFoam` solver was modified to remove all computations of primitive variables. For assessment purposes, the fiber orientation evolution results obtained in OpenFOAM® were compared with reference data obtained from the `Python` scripts provided in the repository folder `fiberOrientationPython`, which integrate the same fiber orientation evolution equations assuming homogeneous time evolving flows. For that purpose, a Runge-Kutta5(4) time integration scheme [27] was used, with an absolute and relative tolerances of 10^{-12} for both quantities.

7.4.1 Simple-shear case study

The velocity field for a simple-shear flow, with velocity along x, and a null velocity at $y = 0$ plane, is given by:

$$\mathbf{v} = \begin{bmatrix} \dot{\gamma} y \\ 0 \\ 0 \end{bmatrix}, \tag{7.1}$$

which, in accordance with the definition of gradient in OpenFOAM®, will result in the velocity gradient:

$$\nabla \mathbf{v} = \begin{bmatrix} 0 & 0 & 0 \\ \dot{\gamma} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{7.2}$$

To verify the accuracy of the numerical implementation, a mesh refinement study using 7 progressively refined grids was performed on the three-dimensional geometry shown in Figure 7.2. The coarser mesh has 32 cells along the x- and y-directions and one cell in the z- direction. The number of cells in the x-direction were doubled for each successively finer grid.

The initial velocity field and boundary conditions were defined with the utilities `setExprFields` and `setExprBoundaryFields` by applying the exact solution in Equation (7.1). The shear-rate and fiber shape factor were assumed to be unitary, $\dot{\gamma} = 1\, s^{-1}$ and $\xi = 1$. The initial and Dirichlet boundary conditions for A_{ij} on the `left` patch is given by:

$$A_{ij} = \begin{bmatrix} 1/3 & 0 & 0 \\ 0 & 1/3 & 0 \\ 0 & 0 & 1/3 \end{bmatrix}. \tag{7.3}$$

For the "(`right` | `top` | `bottom`)" patches a `zeroGradient` condition was assigned and for the "(`front` | `back`)" patches a `symmetryPlane` boundary condition was used.

For this case study, three fiber orientation models were tested, namely FT, iARD-RPR and MRD. The IBOF closure relation was employed with all models. For the FT model, two additional closure relations were tested, Hybrid and ORE. The relevant parameters for these models and closure relations are reported in Table 7.1 and were obtained from Favaloro and Tucker [28].

The discretization schemes and solution requirements employed are reported in Table 7.2 and 7.3, respectively. The orientation evolution equation was integrated from 0 to 65 s, with the time step size being controlled by the CFL condition for a maximum Courant number of 0.5.

Table 7.1: Fiber orientation model parameters for the simple-shear case study.

	Model		
Parameter	FT	iARD-RPR	MRD
C_I	0.0311	0.0562	0.0198
C_M	–	0.9977	–
α	–	0	–
D_i	–	–	$D_2 = 0.4796$ $D_3 = 0.0120$
Closure	Hybrid IBOF ORE	IBOF	IBOF

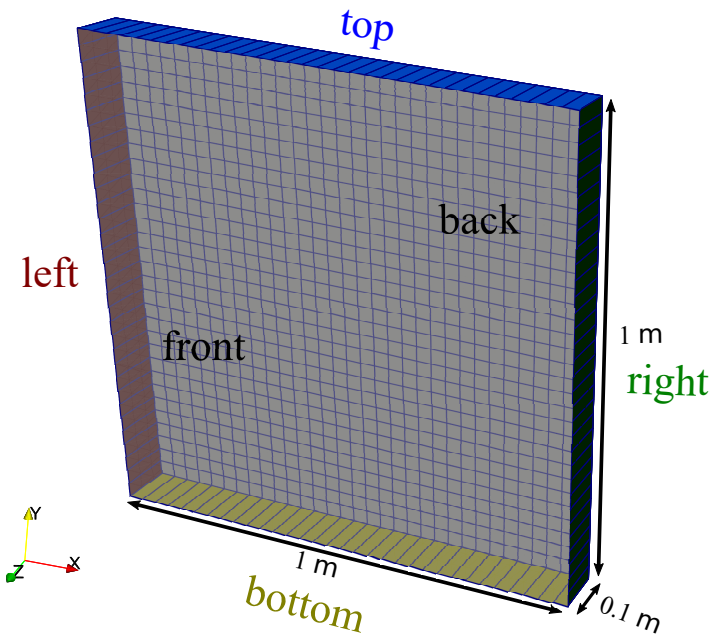


Figure 7.2: Geometry, mesh and patches for the simple-shear case study.

Table 7.2: Temporal and spatial discretization schemes for the simple-shear case study.

Operator / term	Discretization scheme	OpenFOAM® terminology
Time derivative, $\partial\phi/\partial t$	implicit Euler	`Euler`
Gradient, $\nabla\mathbf{u}$	central differencing	`Gauss linear`
Convective term, $\nabla\cdot(\phi\, A_2)$	first-order upwind	`Gauss upwind`
Laplacian, $\nabla\cdot(\Gamma\nabla\phi)$	not used	`none`
Interpolation from cell centers to faces	linear interpolation	`linear`
Surface-normal gradient, $\nabla_f^{\perp}\phi$	not used	`none`

Table 7.3: Solution methods for the simple-shear case study.

Parameter	
solver	`PBiCGStab`
smoother	`DILU`
tolerance	10^{-10}
relative tolerance	0

7.4.2 Center-gated disk case study

The center-gated disk is a representative benchmark case of a typical geometry in the injection-molding process. Due to the usual small thickness of parts produced by IM, the flow is typically dominated by shear, but extensional flow can also occur [29]. In this case study, assuming constant viscosity (Newtonian behavior) is not realistic for IM of thermoplastic materials. However, it was used for convenience in verifying the implemented fiber orientation add-on, as an analytical solution is available.

The geometry and patches considered for the present case study are illustrated in Figure 7.3. Due to the axisymmetric flow conditions just a small angular region of the disk was employed for verification purposes. Accordingly, the geometry in this case study is a $5°$ slice from a typical center-gated disk with a feeding system, with a radius $r \in [0.01, 0.12]$ m and a thickness of 0.003 m.

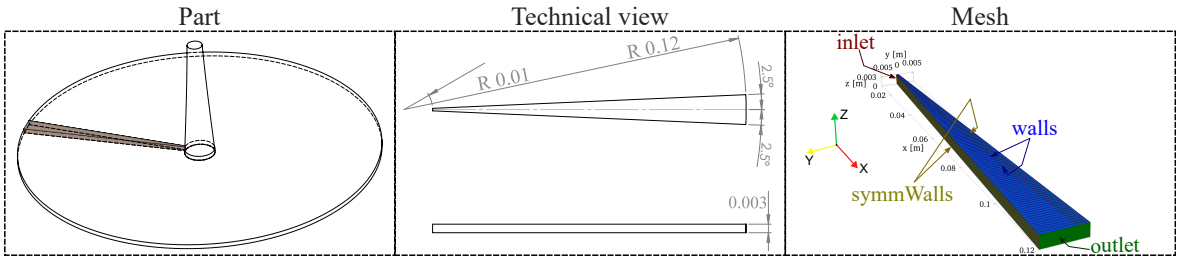


Figure 7.3: Part and mesh representation with the corresponding patches for the center-gated disk case study (dimensions in m).

The velocity field, in cylindrical coordinates, for a Newtonian fluid in a center-gated disk is given by Equation (3.18) with the velocity gradient, following the definition of OpenFOAM®, given as the transpose of Equation (3.19).

Due to the symmetry of the problem and the Cartesian gradient implemented in OpenFOAM®, the geometry was defined so that the x-direction aligns with the radial direction, $y = 0$, and z is perpendicular to the flow direction. For verification purposes a mesh refinement study was carried out, using five progressively finer meshes. The coarser mesh has 128 cells along the radial and thickness directions and one along the angular direction (θ in the cylindrical coordinates). The cells in the radial direction were doubled for each successively finer grid to increase the accuracy of the fiber orientation evolution.

For this case study, the flux and velocity fields were computed assuming a total flow rate of $Q = 134.774\ \mathrm{cm^3 s^{-1}}$. Given that only a $\alpha = 5°$ slice is being considered, the case study flow rate was adjusted to $Q\alpha/360$.

For this case, the flux and the gradient at the cell center were prescribed through a coded `functionObject`. For the second-order orientation tensor, the initial condition and Dirichlet boundary condition for the inlet were defined as isotropic state, Equation (7.3). A `zeroGradient` boundary condition was applied to the `walls` and `outlet`, and a `symmetry` condition was applied to the `symmWalls` patch. The discretization and solution settings were the same as those considered for the simple-shear case study. Moreover, the fiber orientation parameters are described in Table 7.4.

A `coded functionObject` was used to define a steady-state criterion. The simulation was stopped when the initial maximum residual of A_{ij} was below 10^{-6}, and during 15 consecutive time steps the absolute difference between the current and previously computed A_{ij} was below 10^{-6}, in all computational cells.

Table 7.4: Fiber orientation model parameters for the center-gated disk case study.

	Model
Parameter	FT
ξ	1
C_I	0.001
closure	IBOF

7.5 Results and discussion

7.5.1 Simple-shear case study

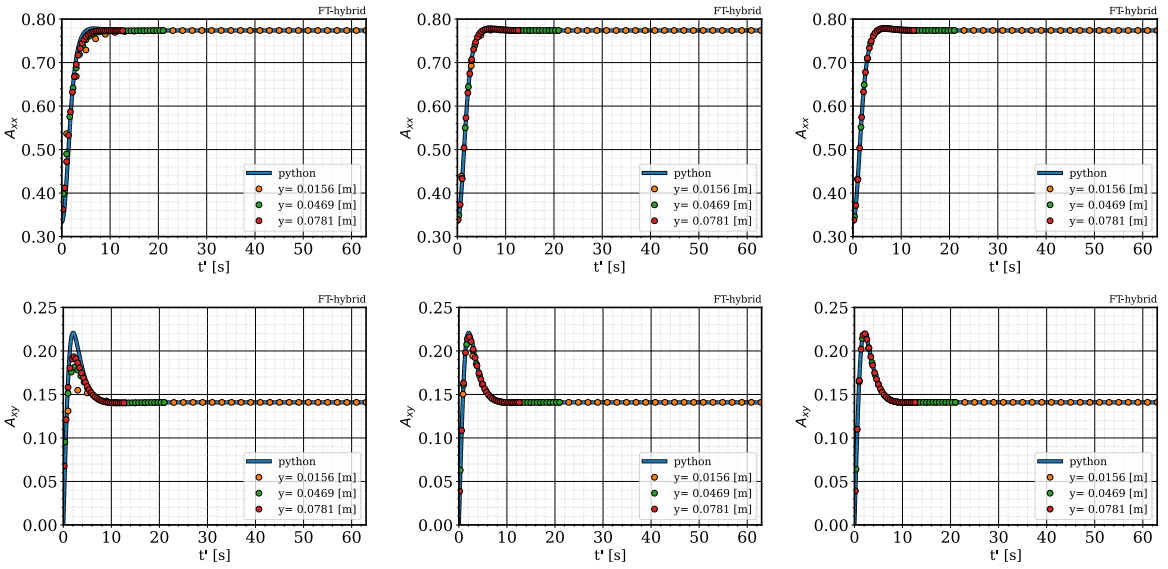
In the simple-shear case study, taking into account the velocity field given by Equation (7.1), the A_{xz} and A_{yz} components of the fiber orientation tensor are null, whereas the A_{zz} component will have a similar dynamic to the A_{xx} component [30]. Thus, for verification purposes, only the A_{xx} and A_{xy} components will be analyzed. The A_{xx} component indicates the alignment in the x-direction and the off-diagonal component the direction of orientation in the $x-y$ plane [30]. To compare the results from the `Python` script with the ones computed in OpenFOAM®, and given that all streamlines are parallel to the x-axis, a pseudo-time ($t^{'}$) was calculated based on the cell velocity and the domain length by:

$$t^{'} = \frac{L_x}{u}, \tag{7.1}$$

where L_x is the length of the domain along the x-direction, with $L_x = 0$ at the inlet, and u is the

x-direction velocity component.

Given that only the velocity component in the x-direction is non-null, each line of cells in the x-direction is expected to match a section in the fiber orientation curve. Figures 7.4, 7.5 and 7.6 depict the comparison between the A_{xx} and A_{xy} components of the fiber orientation tensor with progressive mesh refinement for the FT model using the different closure models tested. The scatter data displayed corresponds to the line of cells at the y-position with the most significant numerical difference compared with the `Python` reference data. Figure 7.7 shows the evolution of A_{xx} when using the iARD and MRD models with the IBOF closure.

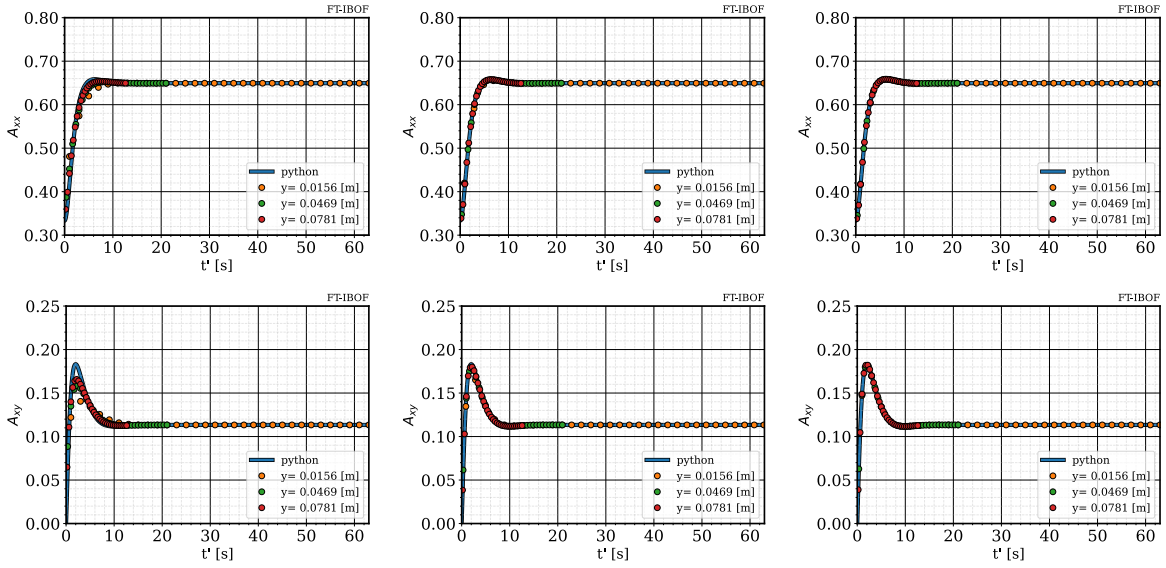


(a) Mesh 1 (b) Mesh 4 (c) Mesh 7

Figure 7.4: Mesh refinement study with the Folgar-Tucker model and the Hybrid closure for the simple-shear case study.

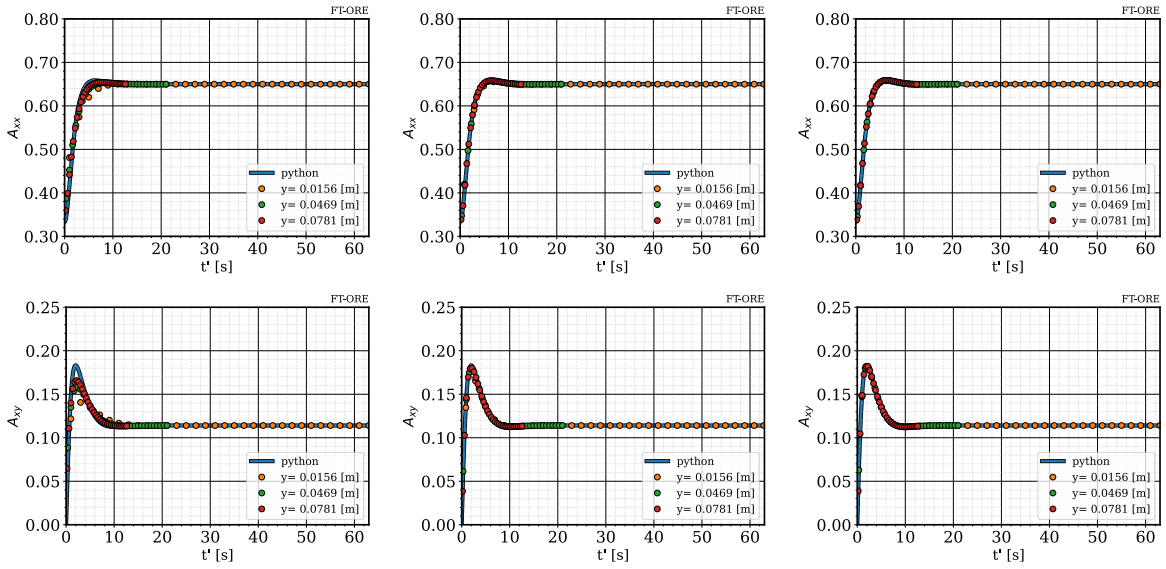
The parameters defined for the models were set to obtain roughly the same value at a steady-state ($A_{xx} \approx 0.65$) [28]. Comparing the different closures used with the FT model it is clear that the hybrid model over-predicts the value of orientation for this simple-flow. This is a known issue in literature [30, 31], however, due to its computational simplicity it is employed in some proprietary software. The IBOF and ORE present very similar transient results. This has also been addressed in literature [31], where the major practical difference is the computational effort. The ORE requires the eigendecomposition of the tensor A_{ij} whilst the IBOF works with its invariants. Comparing the FT model with the iARD and MRD, although the solution at steady-state is the same, the transient behavior differs substantially. The FT model flattens out after ≈ 10 s while the iARD and MRD model present a pronounced peak. This shows the ability of different phenomenological orientation models to capture different orientation behaviors, whose choice is not trivial.

The results also show that with progressive mesh refinement, the simulated data computed in OpenFOAM® approximates the reference data generated with the `Python` script.



(a) Mesh 1 (b) Mesh 4 (c) Mesh 7

Figure 7.5: Mesh refinement study with the Folgar-Tucker model and the IBOF closure for the simple-shear case study.

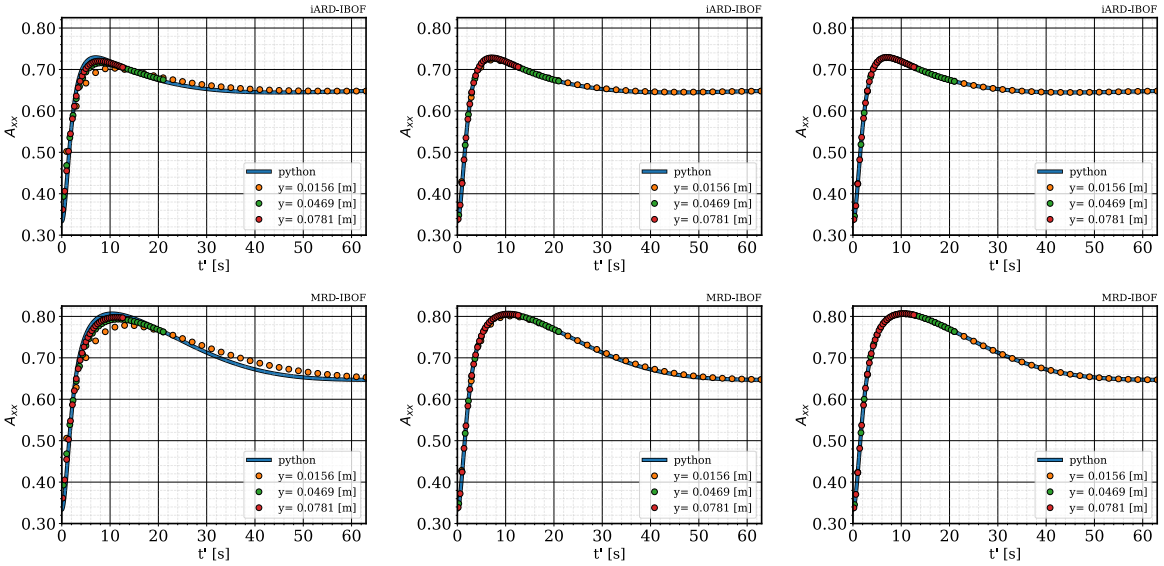


(a) Mesh 1 (b) Mesh 4 (c) Mesh 7

Figure 7.6: Mesh refinement study with the Folgar-Tucker model and the ORE closure for the simple-shear case study.

7.5.2 Center-gated disk case study

The parabolic flow profile given by Equation (3.18) represents a non-homogeneous radial diverging flow field where the velocity magnitude decreases along the radial direction, and, accordingly, the shear ($\dot{\gamma}$) and stretch ($\dot{\epsilon}$) rates ratio are functions of the thickness and radial position,



(a) Mesh 1 (b) Mesh 4 (c) Mesh 7

Figure 7.7: Mesh refinement study with the iARD and MRD models for the A_{xx} component for the simple-shear case study

as illustrated in Figure 7.8.

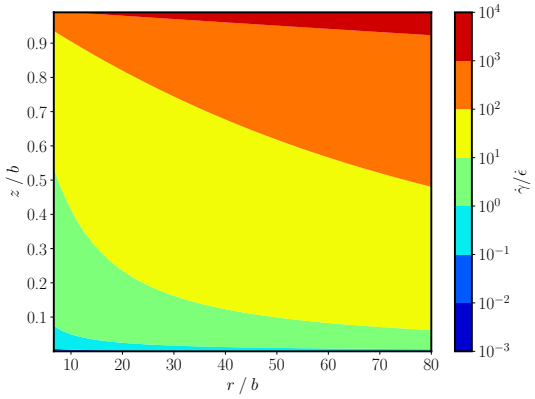


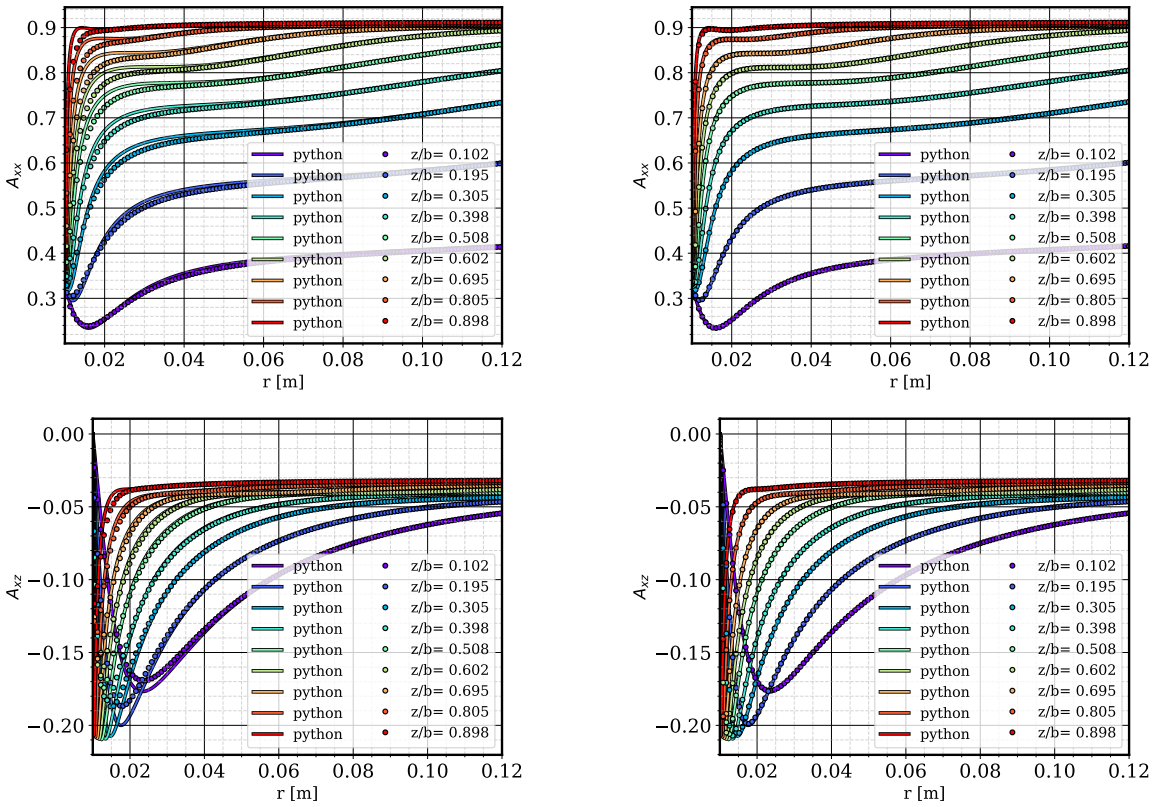
Figure 7.8: Map of shear/stretch ratio for the center gated disk.

Figure 7.8 depicts the ratio between the shear and extensional components on half-thickness of the disk. For these conditions, null shear and elongation will occur at the mid-plane and wall, respectively. At the mid-plane, where $\frac{\partial u}{\partial x} < 0$ with null shear-rate, the fibers are compelled to orient along the cross-flow direction. As the magnitude of z increases, the shear-rate becomes more pronounced, reaching its maximum at the wall. Consequently, the fibers tend to orient increasingly along the flow direction.

Similarly to the simple-shear case study, the A_{xy} and A_{yz} components of the fiber orientation tensor are always zero, whereas the A_{zz} component will have a similar dynamic to the A_{xx} (since the trace must be unitary). Thus, for verification purposes, only the A_{xx} and A_{xz} components will be

presented.

Referring to the results shown in Figure 7.8 and Figure 7.9, near the mid-plane, at $z/b = 0.102$, the A_{xx} component initially tends towards zero due to the presence of a negative elongational rate and an insufficient shear-rate to keep the alignment of the fibers in the flow direction. However, as the fluid advances along the radial direction, the small but non-null shear-rate gradually, albeit slowly, aligns the fibers along the flow direction. This effect intensifies with z/b, i.e., while approaching the wall.



(a) Mesh 1

(b) Mesh 5

Figure 7.9: Mesh refinement study with the Folgar-Tucker model and the IBOF closure for the center-gated disk case study.

To compare the results obtained in OpenFOAM®, the FT model was integrated using 3000 linearly spaced points along the radius, and 128 points along the thickness corresponding the z-coordinate of the mesh. The orientation tensor was sampled at the cell centers with the `sampling functionObject` utility of OpenFOAM®. The mesh refinement results for the closest ten evenly spaced z/b ratios are displayed in Figure 7.9. The mesh refinement study shows that with progressive refinement, the results obtained with OpenFOAM® approach the reference data generated from the `Python` script.

7.6 Conclusions

In this work, a `functionObject` that works as a plug-in tool for any OpenFOAM® incompressible flow solver was programmed to calculate the evolution of fiber orientation in fiber reinforced thermoplastic materials, following a decoupled approach. Several fiber orientation models and closure relations available in the literature were implemented in the tool, which was verified by comparing the solver predictions with independent results obtained by numerically integrating the associated governing equations.

Due to OpenFOAM®'s limitations in handling higher-order tensors, the developed methodology for modeling fiber orientation, which requires fourth-order tensors, utilizes open-source symbolic computation to perform the necessary calculations. This symbolic approach retains only the non-trivial operations, thus enhancing efficiency. However, this comes with the cost of code readability, as the functions resulting from the symbolic computations tend to be almost impossible to interpret.

State-of-the-art models and closure relationships currently employed in commercial solutions have been described and implemented in OpenFOAM®. Benchmark test cases, including a simple-shear flow and the flow in a center-gated disk, have been used to verify the numerical implementation, through the comparison with results obtained with a `Python` routine that integrates the equations for a single material point. A mesh refinement study was carried out for each type of flow. The results show that with progressive mesh refinement the OpenFOAM® results approach the `Python` counterpart.

The developed code, scripts, and case studies are open to the community, and the authors welcome suggestions and improvements.

Appendix

Availability of data and material

The data that support the findings of this study are openly available at: `https://github.com/Computational-Rheology/paper_OFJ_FiberOrientation`

References

[1] Chao-Tsai Huang and Huan-Chang Tseng. Simulation prediction of the fiber breakage history in regular and barrier structure screws in injection molding. *Polymer Engineering & Science*, 58(4):452–459, 2018.

[2] F. W. J. Van Hattum and C. A. Bernardo. A model to predict the strength of short fiber composites. *Polymer Composites*, 20(4):524–533, 1999.

[3] M. Sepehr, G. Ausias, and P.J. Carreau. Rheological properties of short fiber filled polypropylene in transient shear flow. *Journal of Non-Newtonian Fluid Mechanics*, 123(1):19–32, 2004.

[4] Seyyedvahid Mortazavian and Ali Fatemi. Effects of fiber orientation and anisotropy on tensile strength and elastic modulus of short fiber reinforced polymer composites. *Composites Part B: Engineering*, 72:116–129, 2015.

[5] Davide Masato, Jitendra Rathore, Marco Sorgato, Simone Carmignato, and Giovanni Lucchetta. Analysis of the shrinkage of injection-molded fiber-reinforced thin-wall parts. *Materials & Design*, 132:496–504, 2017.

[6] Yue Mu, Guoqun Zhao, Anbiao Chen, Yifan Liu, Yining Song, and Chao Sun. Numerical investigation of three-dimensional fiber suspension flow by using finite volume method. *Polymer Bulletin*, 74(11):4393–4414, 2017.

[7] Suresh G. Advani and III Tucker, Charles L. The Use of Tensors to Describe and Predict Fiber Orientation in Short Fiber Composites. *Journal of Rheology*, 31(8):751–784, 11 1987.

[8] Jin Wang, John F. O'Gara, and III Tucker, Charles L. An objective model for slow orientation kinetics in concentrated fiber suspensions: Theory and rheological evidence. *Journal of Rheology*, 52(5):1179–1200, 09 2008.

[9] Jay H. Phelps and Charles L. Tucker. An anisotropic rotary diffusion model for fiber orientation in short- and long-fiber thermoplastics. *Journal of Non-Newtonian Fluid Mechanics*, 156(3):165–176, 2009.

[10] Huan-Chang Tseng, Rong-Yeu Chang, and Chia-Hsiang Hsu. An objective tensor to predict anisotropic fiber orientation in concentrated suspensions. *Journal of Rheology*, 60(2):215–224, 03 2016.

[11] Huan-Chang Tseng, Rong-Yeu Chang, and Chia-Hsiang Hsu. The use of principal spatial tensor to predict anisotropic fiber orientation in concentrated fiber suspensions. *Journal of Rheology*, 62(1):313–320, 01 2018.

[12] A Bakharev, H Yu, S Ray, R Speight, and J Wang. Using new anisotropic rotational diffusion model to improve prediction of short fibers in thermoplastic injection molding. In *SPE ANTEC Conference*, 2018.

[13] Susanne Katrin Kugler, Argha Protim Dey, Sandra Saad, Camilo Cruz, Armin Kech, and Tim Osswald. A flow-dependent fiber orientation model. *Journal of Composites Science*, 4(3), 2020.

[14] George Barker Jeffery. The motion of ellipsoidal particles immersed in a viscous fluid. *Proc. R. Soc. Lond. A*, 102:161–179, 1922.

[15] Mahesh Gupta and K. K. Wang. Fiber orientation and mechanical properties of short-fiber-reinforced injection-molded composites: Simulated and experimental results. *Polymer Composites*, 14(5):367–382, 1993.

[16] Witold Ogierman and Grzegorz Kokot. A study on fiber orientation influence on the mechanical response of a short fiber composite structure. *Acta Mechanica*, 227(1):173–183, 2016.

[17] Huan-Chang Tseng, Rong-Yeu Chang, and Chia-Hsiang Hsu. Numerical prediction of fiber orientation and mechanical performance for short/long glass and carbon fiber-reinforced composites. *Composites Science and Technology*, 144:51–56, 2017.

[18] Kerstin Heinen. *Mikrostrukturelle Orientierungszustände strömender Polymerlösungen und Fasersuspensionen*. PhD thesis, Universität Dortmund, 2007.

[19] Felix Ospald. *Contributions to the Simulation and Optimization of the Manufacturing Process and the Mechanical Properties of Short Fiber-Reinforced Plastic Parts*. PhD thesis, Technische Universität Chemnitz, 2017.

[20] Florian Wittemann, Robert Maertens, Alexander Bernath, Martin Hohberg, Luise Kärger, and Frank Henning. Simulation of reinforced reactive injection molding with the finite volume method. *Journal of Composites Science*, 2(1), 2018.

[21] Florian Wittemann, Robert Maertens, Luise Kärger, and Frank Henning. Injection molding simulation of short fiber reinforced thermosets with anisotropic and non-Newtonian flow behavior. *Composites Part A: Applied Science and Manufacturing*, 124(May):105476, 2019.

[22] Kunji Chiba and Francisco Chinesta. Numerical simulation of flow kinematics and fiber orientation for multi-disperse suspension. *Rheologica Acta*, 45(1):1–13, 2005.

[23] M. Oliveira. Development of a utility in OpenFOAM to predict fiber orientation in fiber reinforced thermoplastic materials. Master's thesis, University of Minho, 2023.

[24] C.L. Tucker. *Fundamentals of Fiber Orientation: Description, Measurement and Prediction*. Hanser Publishers, 2022.

[25] *SymPy*. `https://www.sympy.org/en/index.html`. Accessed: 22024-02-27.

[26] Bruno Ramoa, Ricardo Costa, Francisco Chinesta, and J. Miguel Nóbrega. A semi-automatic approach based on the method of manufactured solutions to assess the convergence order in openfoam. *OpenFOAM® Journal*, 2:148–165, Nov. 2022.

[27] Scipy user manual. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html`. Accessed: 2024-02-27.

[28] Anthony J. Favaloro and Charles L. Tucker. Analysis of anisotropic rotary diffusion models for fiber orientation. *Composites Part A: Applied Science and Manufacturing*, 126:105605, 2019.

[29] Rong Zheng, Roger I. Tanner, and Xi-Jun Fan. *Injection Molding: Integration of Theory and Modeling Methods*. Springer-Verlag Berlin Heidelberg, 2011.

[30] Suresh G. Advani and Charles L. Tucker. Closure approximations for three□dimensional structure tensors. *Journal of Rheology*, 34(3):367–386, 1990.

[31] Du Hwan Chung and Tai Hun Kwon. Invariant-based optimal fitting closure approximation for the numerical prediction of flow-induced fiber orientation. *Journal of Rheology*, 46(1):169–194, 2002.

Chapter 8 A unifying open-source numerical framework for modeling injection molding of thermoplastic materials

Abstract:

Given the widespread use of Injection Molding (IM) for manufacturing plastic parts, its numerical modeling has been the focus of extensive research in both academia and industry. However, proprietary technologies are often presented as black-box solutions, lacking transparency in which governing equations are actually being solved, which discretization schemes are used and what solution strategies are employed. To establish a transparent and extensible framework for modeling the IM process, this work presents an open-source toolbox named `IMFoam`, developed within the community branch of OpenFOAM®. This toolbox consolidates and extends state-of-the-art capabilities for the filling stage under both incompressible and compressible fluid flow formulations. It employs a single-fluid formulation using the Volume-of-Fluid (VoF) method for interface capture, offering two algebraic VoF formulations and one geometrical formulation. Additionally, it includes multiple pressure-velocity coupling strategies, along with dynamic outlet boundary conditions to more accurately represent the filling dynamics.

The correctness of the discretization of the governing equations and constitutive models is verified through the Method of Manufactured Solutions. A pure-advection test benchmark, along with representative two- and three-dimensional mold filling studies, are conducted to evaluate interface quality, boundedness, and sensitivity to wall/outlet treatments under incompressible assumptions. The results indicate that the geometric variant exhibits sharper and more robust interface retention, albeit at an increased computational cost. The dynamic outlet conditions may lead to mass loss and backflow during the mold filling process, unless additional constraints are enforced on the phase indicator transport equation. Finally, `IMFoam` is validated against a widely used commercial solver by comparing the mold filling progression, inlet pressure history, and velocity/pressure evolution at discrete locations in the mold cavity. The results predicted by `IMFoam` are in reasonable agreement with the commercial counterpart. However, the results from the commercial solver are considered a reference rather than ground truth, due to the lack of a formal grid convergence and the presence of non-physical artifacts, especially in the pressure history. The comparative results are discussed

This chapter was adapted from Ramoa, B., Chinesta, F., & Nóbrega, J. M. A unifying open-source numerical framework for modeling Injection Molding of thermoplastic materials submitted to Polymer Science and Engineering (2026)

qualitatively, showing that computational performance is a key area for future optimization.

Keywords: Injection molding; Numerical modeling; Fiber orientation modeling

8.1 Introduction

Injection molding (IM) is one of the most widespread technologies for manufacturing thermoplastic parts due to its high throughput, design flexibility, and dimensional accuracy. This manufacturing technique consists of a cyclic process in which molten material is injected into a molding cavity with a predefined geometry. The material is then held under pressure to compensate for any volumetric shrinkage while heat is removed from the part through the mold walls and its cooling channels. Once the part has developed sufficient mechanical strength, the molding tool opens, and the molding cycle concludes with the ejection of the part(s) [1].

The high costs associated with the development of manufacturing tools make the use of CAD/CAM/CAE technologies essential for improving time-to-market, design robustness, process optimization, and the prediction of final part properties. Proprietary software [2–6] to model the IM process typically decomposes the modeling workflow into three stages: i) filling, ii) packing, and iii) cooling. These tools enable the prediction of several quantities of interest, including velocity, pressure, and temperature evolutions, as well as other quantities of interest, such as fill pattern, fiber orientation, and molding related defects such as sink marks and weld-lines, among others, thereby supporting more informed and reliable design and production decisions.

Although these technologies are widely employed in industrial practice, they are typically provided as black-box tools. Users have little to no access to the underlying source code, which severely restricts the implementation of new governing equations, thermophysical models, boundary conditions, solution strategies, and independent verification and scientific scrutiny.

Open-source frameworks for computational fluid dynamics provide an attractive path toward transparent and extensible modeling of manufacturing processes. Notably, the OpenFOAM® [7] ecosystem, which has been extensively utilized to model a wide range of manufacturing technologies [8–11] related to thermoplastic materials processing.

Within the OpenFOAM® framework, several studies have addressed the challenges of modeling the IM process filling stage using either `interFoam` [12–14] or `compressibleInterFoam` [8, 15], for incompressible and compressible formulations, respectively, using a one-fluid algebraic Volume-of-Fluid (VoF) formulation. More recently, geometric VoF variants [16–18], which allows sharper interface advection, have also been reported within incompressible formulations [19]. In these studies, existing solvers are augmented with adequate constitutive models and equations-of-state.

Nevertheless, specific modeling challenges persist, particularly regarding the treatment of wall boundary conditions that ensure physically consistent filling dynamics. A commonly employed strategy is the so-called ”dynamic outlet”, which is a boundary condition that toggles between traction-free and impermeable wall conditions based on the local phase-fraction [20, 21]. While conceptually convenient for complex geometries, this approach can induce polymer mass loss and allow backflow into the

domain, thereby introducing an unintended phase mixture through the walls. To simplify the wall conditions, authors have been adopting compressible formulations to be able to use impermeable conditions at the mold walls [8, 15]. However, assuming that the polymer melt can freely displace the air out of the domain during filling, depending on the complexity of the geometry, the use of impermeable conditions can lead to non-physical filling dynamics [22].

This work builds on the contribution of Pedro et al. [8] by consolidating, enhancing, and extending numerical tools for modeling the IM process under both incompressible and compressible formulations, while also integrating relevant state-of-the art algorithms and numerical approaches. The developed utility, named `IMFoam`, was created in the community-driven fork of OpenFOAM, foam-extend [23]. This toolbox offers a selection between algebraic and geometric VoF methods, supports multiple pressure–velocity coupling algorithms, and introduces dynamic outlet boundary conditions to more accurately capture the filling dynamics of the IM process.

To assess the correctness of the implemented solution procedure, the discretization of the governing equations, including generalized Newtonian constitutive models and a suitable equation-of-state for the IM process, is verified using the Method of Manufactured Solutions [24] in conjunction with the SIMPLEC pressure–velocity coupling algorithm. To compare the performance of the available VoF variants in the toolbox, a pure advection test based on a widely used two-dimensional benchmark case study [16] in conducted. In this test case, a bubble is deformed by the influence of a vortical flow, allowing for a comparison of the available phase advection algorithms. A comparative analysis of numerical strategies employing different dynamic outlet conditions is conducted using two- and three-dimensional geometries to investigate the developed permeable boundary conditions under incompressibility assumptions. Lastly, a case study comparing the performance of the developed solver against a proprietary counterpart has been carried out on laboratory specimens to further validate its efficacy.

The remainder of the manuscript is organized as follows: Section 8.2 presents the mathematical formulation of the injection molding process together with its solution strategy. Section 8.3 details the numerical tests conducted to verify the accuracy of the numerical tool and algorithmic validation procedures. Section 8.6 discusses the validation studies on laboratory specimens and compares its performance with a proprietary software specialized in modeling the IM process. Finally, Section 8.7 summarizes the main conclusions of this work and outlines perspectives for future research.

8.2 Mathematical formulation

This section outlines the governing equations for the various stages of the IM process, together with appropriate equations-of-state and rheological models.

8.2.1 Filling

The filling stage, despite being the shortest in duration, is the most critical and complex part of the entire cycle. Several important thermophysical phenomena occur during this stage, along with filling related defects such as weld-lines, air entrapment, among others. Modeling this stage in the IM cycle can be characterized as a three-dimensional, multiphase, non-isothermal, non-Newtonian problem. Inside the cavity, only polymer and air are present, allowing the filling stage to be modeled using either two-fluids [25, 26] or one-fluid [21] methodologies. The former models the process by defining two sets of Navier-Stokes equations coupled with a source term to account for changes in mass, momentum, and energy between phases, while the latter considers a single fluid with properties described by a weighting function based on a local phase indicator. The remainder of this discussion will be focused on the one-fluid formulation.

Given the immiscibility of the polymer melt and air phases, the governing equations for filling stage are the standard continuity, momentum balance and energy conservation equations, along with a phase advection equation, as detailed in [1, 27]. These equations can be expressed as:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{8.1}$$

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = \nabla \cdot \boldsymbol{\tau} - \nabla p + \mathbf{F}_s + \rho \mathbf{g}, \tag{8.2}$$

$$\frac{\partial \left(\rho C_p T\right)}{\partial t} + \nabla \cdot \left(\rho C_p \mathbf{u} T\right) = \beta T \left(\frac{\partial \rho}{\partial t} + \mathbf{u} \cdot \nabla p\right) + p \left(\nabla \cdot \mathbf{u}\right) + \boldsymbol{\sigma} : \nabla \mathbf{u} + \nabla \cdot \left(k \nabla T\right), \tag{8.3}$$

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha \mathbf{u}) = \alpha \left(\nabla \cdot \mathbf{u}\right) + \alpha \left(1 - \alpha\right) \left(\frac{\psi^g}{\rho^g} - \frac{\psi^p}{\rho^p}\right) \frac{\mathrm{D}p}{\mathrm{D}t}, \tag{8.4}$$

with ρ, $\mathbf{u}$, p, T and α being the density, velocity vector, pressure, temperature and volume fraction indicator, respectively. The parameters, C_p, β, k, ψ represent, the specific heat capacity, the coefficient of volume expansion, the thermal conductivity and compressibility of the material, respectively. In addition, $\boldsymbol{\sigma}$, $\mathbf{F}_s$ and $\mathbf{g}$ are the total stress tensor (expressed as $\sigma = -p\mathbf{I} + \boldsymbol{\tau}$, with $\boldsymbol{\tau}$ being the viscous stress tensor), the surface force and gravity acceleration, respectively. In Equation (8.4), the subscripts p and g denote the polymer and gas phases, respectively.

Due to the single set of equations, a generic thermophysical property (Φ) of the immiscible mixture is computed via a volume fraction weighted average, as:

$$\Phi_{\text{mixture}} = \alpha \, \Phi_{\text{polymer}} + (1 - \alpha) \, \Phi_{\text{air}}. \tag{8.5}$$

In the literature, the modeling of the filling stage has been carried out under various assumptions, including incompressibility [13, 25, 28–31], isothermal [25, 26, 32] (justified by the short duration of the filling stage), and compressible formulations [21, 33–35]. The incompressibility assumption is

a feasible approximation for the most thermoplastic polymers used in IM, during most of the filling stage [36]. However, a more accurate description can be achieved, especially towards the end of filling, by considering a compressible formulation [36]. Additionally, thermal properties such as C_p and k, which are temperature and pressure dependent, are commonly held as constants [36]. This stage typically lasts until $95 - 99\%$ part filling by volume.

An interesting aspect of this stage is the treatment given to the air phase, which is often considered unimportant to the overall analysis. In IM simulations, to improve numerical stability, the air is frequently treated as a pseudo-fluid (pseudo-air), with its viscosity set to $\approx 10^{-3}$ times that of the polymer [21, 37]. Due to the high polymer melt viscosity, the capillary number is typically high, resulting in viscous forces dominating over surface tension effects. Therefore, the surface-tension force, $\mathbf{F}_s$, can generally be neglected [38]. If the analysis necessitates the inclusion of this term, the Continuum Surface Force method [39] can be utilized to convert the surface force $\mathbf{F}_s$ into a body force, as follows:

$$\int_S \mathbf{F}_s dS \approx \sigma \kappa \nabla \alpha, \tag{8.6}$$

with σ representing a scalar surface tension and κ the interface curvature.

8.2.2 Packing

During the packing stage, a small amount of additional molten material is introduced into the cavity to compensate for volumetric shrinkage [36]. Accurate modeling of this stage requires accounting for melt compressibility. Accordingly, a suitable model is necessary to describe how the polymer density varies with pressure and temperature. For thermoplastic materials, this dependence is expressed through a pressure-volume-temperature (PVT) relationship, with the density computed as the reciprocal of the specific volume (V), i.e., $\rho = 1/V$. This stage significantly contributes to the development of residual stresses in the part [36].

8.2.3 Cooling

Cooling occurs concurrently with the filling and packing phases. However, without significantly compromising the accuracy of the predictions, the cooling stage can be considered to begin when no more molten polymer is flowing into the cavity [1]. At this point, only heat transfer occurs between the polymer and the mold walls, with heat being removed from the mold through the cooling channels, and the only active governing equation being the energy Equation (8.3).

For this stage, typical boundary conditions include: Dirichlet (imposed temperature value), Neumann (imposed temperature gradient) or Robin(e.g., convective heat flux) to compute the temperature evolution in the computational domain. If the analysis of heat transfer to the tool is of

interest (e.g., to optimize the cooling channels layout), a conjugate heat transfer formulation must be employed, by considering the balance of heat fluxes at the mold-polymer interfaces.

8.2.4 Rheological models

Flows developed during injection molding are typically dominated by shear forces, with the behavior of the flow primarily influenced by viscosity. Although thermoplastic materials are viscoelastic in nature, a generalized Newtonian (inelastic) approximation is often used to model the rheological behavior of these materials in IM [1–6]. The constitutive equation for such fluids, accounting for compressibility effects, is given by:

$$\boldsymbol{\tau} = 2\mu\left(\dot{\gamma}, p, T\right)\mathbf{D} - \frac{2}{3}\mu\left(\dot{\gamma}, p, T\right)\left(\nabla\cdot\mathbf{u}\right)\mathbf{I}, \tag{8.1}$$

where $\mathbf{D}$ is the symmetric part of the velocity gradient ($\nabla\mathbf{u}$), computed as:

$$\mathbf{D} = \frac{1}{2}\left(\nabla\mathbf{u} + \left(\nabla\mathbf{u}\right)^{\mathrm{T}}\right). \tag{8.2}$$

To describe the non-Newtonian behavior of thermoplastic materials in IM, the modified Cross-WLF model is among the most commonly used [1] and is expressed as:

$$\eta = \frac{\eta_0}{1 + \left(\frac{\eta_0\dot{\gamma}}{\tau^*}\right)^{(1-n)}}, \tag{8.3a}$$

$$\eta_0 = D_1 \exp\left(-\frac{A_1\left(T - T^*\right)}{A_2 + \left(T - T^*\right)}\right), \tag{8.3b}$$

$$A_2 = \tilde{A}_2 + D_3 p, \tag{8.3c}$$

$$T^* = D_2 + D_3 p, \tag{8.3d}$$

where η_0 is the zero-shear viscosity, τ^* is the critical stress at the transition to shear thinning, n is the power-law index, T^* is the glass transition temperature and $D_1, A_1, \tilde{A}_2, D_3$ are fitted coefficients. Despite being the most commonly used model, other alternatives have also been reported in the literature depending on the type of polymer and the existence of fillers, i.e., Carreau [40], Carreau-Yasuda [41] and Herschel-Bulkley [42] models.

8.2.5 Equation-of-state

The density of thermoplastic materials depends on temperature and pressure, therefore, a suitable equation-of-state is required to model this relationship. The most commonly used model for

the IM of thermoplastic materials is the modified 2-domain Tait PVT model [1, 43, 44], given by:

$$\hat{V}(p,T) = \hat{V}_0(T)\left[1 - C\ln\left(1 + \frac{p}{B(T)}\right) + \hat{V}_t(p,T)\right], \tag{8.1a}$$

$$\hat{V}_0(T) = \begin{cases} b_{1s} + b_{2s}\bar{T}, & \text{if} \quad T \leq T_t \\ b_{1L} + b_{2L}\bar{T}, & \text{if} \quad T > T_t \end{cases}, \tag{8.1b}$$

$$B(T) = \begin{cases} b_{3s}\exp\left(-b_{4s}\bar{T}\right), & \text{if} \quad T \leq T_t \\ b_{3L}\exp\left(-b_{4L}\bar{T}\right), & \text{if} \quad T > T_t \end{cases}, \tag{8.1c}$$

$$\hat{V}_t(p,T) = \begin{cases} b_7\exp\left(b_8\bar{T} - b_9 p\right), & \text{if} \quad T \leq T_t \\ 0, & \text{if} \quad T > T_t \end{cases}, \tag{8.1d}$$

$$\bar{T} = T - b_5, \tag{8.1e}$$

$$T_t = b_5 + b_6 p, \tag{8.1f}$$

$$C = 0.0894, \tag{8.1g}$$

where $b_{1\cdots4\,L}$ and $b_{1\cdots4\,S}$ and $b_{5\cdots9}$ are data fitted coefficients. This model can describe the variation of density for both semi-crystalline and amorphous thermoplastic materials.

8.2.6 Free surface modeling

During the filling stage, accurate tracking of the melt front is of key importance. Several interface-tracking algorithms have been used in IM simulations, including Arbitrary Lagrangian–Eulerian [29], Level-Set [28, 45, 46] and Volume-of-Fluid [21, 47–50].

The VoF methodology is of special interest due to its conservative nature [51, 52]. In most studies related to IM, an algebraic VoF formulation is employed, where the interface is captured via a volume fraction indicator of the primary-phase (α) in the computational mesh. A value of $\alpha = 1$ indicates a cell completely filled with polymer, whereas $\alpha = 0$ denotes a cell completely filled with the secondary phase (usually pseudo-air). Cells with $0 < \alpha < 1$ contain the interface. To minimize smearing caused by numerical diffusion and to maintain the VoF field bounded during the solution of its hyperbolic transport equation, Equation (8.4), several discretization schemes and limiters have been applied to IM simulations, e.g, CICSAM [53], HRIC [54] or MULES [14].

Within the standard OpenFOAM® framework [55] (as opposed to foam-extend [23]), the two main variants of VoF method are implemented: algebraic [51] and geometric [16, 17]. The latter relies on geometric operations to explicitly approximate interface location on arbitrary polyhedral meshes. This VoF variant has traditionally been computationally more expensive than the algebraic counterpart [16]. However, a sharper interface might prove useful for tracking filling related phenomena, such as weld-lines, provided that the computational cost remains manageable.

8.2.7 Boundary conditions

A proper choice of boundary conditions is one of the most critical aspects when modeling the injection molding process. A survey of the literature reveals a wide array of boundary conditions used for modeling the IM process, particularly for wall conditions. A straightforward choice might be to assign a no-slip condition at the wall. However, as discussed in the work of Kim and Turng [22] and Hetu and coworkers [20], assigning a no-slip condition for the velocity should only be made for the filled portion of the cavity, with possible unrealistic solutions arising since the polymer melt front will never touch the wall. This effect will be illustrated later with numerical examples. It is particularly pronounced when the gas phase is modeled as incompressible and can be mitigated by introducing compressibility in the pseudo-air region.

Throughout the literature of IM [19–22, 38, 56–61], particularly when using finite volume methodology with a VoF [19, 21, 56, 57, 60] a commonly used boundary condition is the so-called "dynamic outlet", with a general framework defined as:

$$\begin{cases} \mathbf{u} = 0 \quad \text{if} \quad \alpha \geq \alpha_{\text{crit}} \\ \boldsymbol{\sigma} \cdot \mathbf{n}_f = 0 \quad \text{if} \quad \alpha < \alpha_{\text{crit}} \end{cases} . \tag{8.1}$$

This condition states that a no-slip condition is applied at the wall when α exceeds a certain threshold, while traction-free condition is imposed otherwise. In this context, the traction free, for incompressible flow, can be defined as [62]:

$$\begin{cases} 2\mu \dfrac{\partial u_n}{\partial n} - p = 0 \\ \mu \left(\dfrac{\partial u_t}{\partial n} + \dfrac{\partial u_n}{\partial \tau} \right) = 0 \end{cases} , \tag{8.2}$$

where u_n and u_t are the normal and tangential velocity components, respectively, and, $\frac{\partial}{\partial n}$ and $\frac{\partial}{\partial \tau}$ are the partial derivative along the normal and tangential directions to the face, respectively. Typically, the term $\frac{\partial u_n}{\partial \tau}$ is neglected [62], although the developed toolbox is capable of computing this quantity at the boundary through point least-squares interpolation [63], this contribution will be disregarded for the remainder of the manuscript, unless otherwise specified. Accordingly, assuming that the pressure datum at the flow front is the atmospheric pressure, the traction-free condition can be approximated by imposing a null normal gradient boundary condition for the velocity components and a fixed gauge pressure value of zero.

Conceptually, this strategy allows cells near the wall to be filled by convecting α out of the domain. However, this results in a loss of polymer mass from the computational domain. In addition, while the no-slip condition is inactive, backflow through a boundary face can occur, leading to an unexpected phase mixture (depending on the value of α at the wall) entering the computational domain through the walls. These effects have, to the best of the authors' knowledge, not been explicitly addressed in

prior research.

Other, similar wall boundary conditions have also been reported in the literature, such as: Navier-slip [32] and Robin penalty [37], although straightforward application of the no-slip condition has also been adopted in some studies [15, 64] under compressible formulation. The use of a dynamic outlet is attractive when dealing with complex geometries where the explicit definition of an outlet is not always straightforward.

In the present work, a mixed (Robin-type) velocity–pressure boundary condition is formulated to reproduce the behavior of Equation (8.1) by switching, depending on the value of α at the cell center, between:

- a no-slip condition on velocity and a null normal gradient on pressure, and
- a zero normal gradient on the velocity and fixed value condition on the pressure.

In the context of OpenFOAM®, this boundary condition is defined as [65]:

$$\phi_f = w\phi_{\text{ref}} + (1 - w)\left(\phi_C + \Delta\nabla\phi_{\text{ref}}\right), \tag{8.3}$$

with ϕ_f being the face value, ϕ_{ref} the reference value, ϕ_C the value at the cell center, Δ the face-to-cell distance and w a the weighting factor. This weighting factor facilitates the transition between pure Dirichlet boundary conditions at $w = 1$ and pure Neumann boundary conditions at $w = 0$.

To study the effect of the dynamic outlet, the mixed boundary condition considers three switching criteria for transitioning between Dirichlet and Neumann. In line with the developments of Chang and Yang [21], a dynamic outlet condition is first considered, implemented via a sharp jump, as defined in Equation (8.1) with $\alpha_{\text{crit}} = 0.5$. Additionally, a linear variation between null gradient and no-slip as reported in [15] is examined. It is worth noting that in that work, the authors only applied this boundary condition at the outlet for the velocity components, while forcing pressure to have a null gradient. After some preliminary experiments, the present implementation of the linear transition is complemented by a clipping operation that activates the no-slip condition slightly before α becomes unitary, which is equated as follows:

$$w(\alpha) = \begin{cases} 1, & \text{if } \alpha \geq 0.99, \\ 0, & \text{if } \alpha < 10^{-6}, \\ \alpha, & \text{otherwise.} \end{cases} \tag{8.4}$$

Finally, a sigmoidal function is adopted to control the smoothness of the transition between both states, following:

$$w(\alpha) = \frac{1}{2} + \frac{1}{2}\tanh\left(\left(\alpha - \alpha_{\text{ref}}\right)k_{\text{sig}}\right), \tag{8.5}$$

where α_{ref} defines the center of the sigmoidal transition and k_{sig} is a scaling factor controlling the sharpness of the transition. Unless otherwise stated, $\alpha_{\mathrm{ref}} = 0.5$ and $k_{\mathrm{sig}} = 20$ are used whenever this boundary condition is applied. Additionally, since Equation (8.5) does not satisfy the endpoint values $w(0)$ and $w(1)$ for $\alpha = 0$ and $\alpha = 1$, respectively, a normalized version is defined as:

$$w\left(\alpha\right)_{\mathrm{norm}} = \frac{w\left(\alpha\right) - w\left(0\right)}{w\left(1\right) - w\left(0\right)}. \tag{8.6}$$

For the sake of completeness, the remaining boundary conditions employed are briefly summarized. At the inlet, a Dirichlet boundary condition is typically assigned for the velocity and temperature, together with a null normal gradient condition for the pressure.

At the walls, the temperature boundary condition is usually defined by either a Dirichlet boundary or a Robin boundary condition representing the convective heat-flux [8].

8.2.8 Solution strategy

Several numerical approaches have been used discretize and solve the IM governing equations, namely: the Finite Element Method [20, 33], the Finite Volume Method (FVM) [21, 49, 57] and Meshless methods [66]. This work focus on the collocated FVM that is the main discretization technique available in the OpenFOAM® framework. The interested reader is refereed to [67] for more information about the per-operator discretization procedure.

In FVM, a computational domain is comprised of non-overlapping polyhedral volumes (computational cells) that share a single face, which approximates the geometrical domain of interest. The associated governing equations are then integrated in each computational cell. By applying the Gauss divergence theorem, the volume integrals of certain differential operators are converted into surface integrals over the boundaries of each cell. This allows for the formulation of linear algebraic equations for each computational cell, relating its unknowns to those of its neighboring cells. The resulting set of algebraic equations form a system of equations whose solution yields the cell-centered unknowns.

In the developed code, pressure-velocity coupling is ensured using a segregated algorithm that combines the Semi-Implicit Method for Pressure-Linked Equations [68] (SIMPLE) and the Pressure-Implicit Split Operator [69] (PISO) algorithms. This combination is commonly referred to as the PIMPLE algorithm [70], and allows multiple outer corrections and the option to apply under-relaxation for enhanced numerical stability. The overall strategy is aimed at alleviating the Courant limitation associated with the PISO algorithm. This methodology is sufficiently flexible to revert to pure SIMPLE or PISO strategies. Additionally, the consistent variant of the SIMPLE algorithm (SIMPLEC) has been implemented alongside the SIMPLE method, as it has demonstrated improved stability for viscoelastic fluid flow simulations [71].

The overall linearization procedure employed, follows the work of Miller et al. [72] but with some modifications, consistent with the standard distributions of `compressibleInterFoam` [73]. A solution flow-chart depicting the solver algorithm is provided in Appendix A.

The standard solution approach begins, after collecting the user input and creating the primary variable fields, with the solution of Equation (8.4) through the chosen VoF algorithm, followed by an update of the mixture properties (e.g. ψ, η, k). The momentum balance and energy conservation equations are then solved sequentially, followed by an update of the thermophysical properties of each phase and the associated mixture properties.

Within the PISO sub-loop, a pressure equation is solved instead of a pressure correction equation [74], in line with other solvers in the OpenFOAM® framework. Depending on the user input, SIMPLEC contributions may also be added to the linearized system of equations [74]. After solving the pressure equation, the volumetric flux is updated, the velocity is corrected, and the density is updated based on the pressure field difference between the previous and current iterations.

Within a time step, the PIMPLE loop is terminated after a predefined number of iterations is reached or when the convergence criteria (absolute and/or relative tolerances) are satisfied.

8.3 Solver verification

8.3.1 Method of manufactured solutions

The method of manufactured solutions (MMS) is a methodology for code verification through quantification of the order-of-accuracy. In this methodology, an analytical solution, referred to as the Manufactured Solution (MS) , with arbitrary complexity is chosen for a governing partial differential equation (PDE) [75]. The MS does not have to be physically realistic. However, for efficiency, it is advised that all terms in the PDE have a similar order of magnitude [76]. The selected solution is introduced into the governing PDE to yield a source term, S_T. Solving the original PDE, with the computed S_T, and appropriate boundary conditions e.g., Dirichlet, Neumann or Robin, that are also consistent with the chosen MS, results in a system of equations whose final solution is the desired MS.

Following the standard MMS approach, the correctness of the numerical code implementation is assessed through a mesh and/or time refinement study that monitors the observed order-of-accuracy. This metric is typically inferred by evaluating the rate at which different global error norms of the discretized problem (typically, L^1, L^2, L^∞) evolve with grid and/or temporal refinement [77, 78]. Assuming a uniform grid, the spatial global error norms are evaluated as follows:

$$e_n = \left|\phi_{\text{exact}_n} - \phi_{\text{FVM}_n}\right|, \tag{8.1}$$

$$\left\|L^1\right\| = \frac{1}{N}\sum_{i}^{N} e_n, \tag{8.2}$$

$$\left\|L^2\right\| = \frac{1}{N}\left(\sum_{i}^{N} e_n^2\right)^{(1/2)}, \tag{8.3}$$

$$\left\|L^\infty\right\| = \max\left(e\right), \tag{8.4}$$

where e_n is the absolute difference between the exact solution and the one computed with the numerical code for the n^{th} cell, and N is total number of cells.

The numerical experiments in Subsections 8.3.1.1 and 8.3.1.2 were carried out on five progressively finer grids, starting with 32 elements along the x and y directions and doubling the number of elements in each Cartesian direction with each successive grid. For all the problems, the aimed orders-of-convergence were computed following the procedure described in Ramoa et al. [24].

8.3.1.1 Viscosity model

The verification of the rheological model implementation was carried out considering the classical Taylor-Green (TG) vortex case [79, 80] in two dimensions. This in an unsteady incompressible flow with decaying vortices that has an exact solution for the incompressible Navier-Stokes equations. Considering a unitary square, a family of solutions is given by Equation (8.5).

$$u = \sin\left(\omega x\right)\,\cos\left(\omega y\right)\,\exp\left(-2\omega\nu t\right), \tag{8.5a}$$

$$v = -\cos\left(\omega x\right)\,\sin\left(\omega y\right)\,\exp\left(-2\omega\nu t\right), \tag{8.5b}$$

$$p = \frac{\rho}{4}\left(\cos\left(2\omega x\right) + \cos\left(2\omega y\right)\right)\exp\left(-2\omega\nu t\right), \tag{8.5c}$$

with ω being the frequency and $\nu = \mu/\rho$ the kinematic viscosity.

The TG solution was considered at $t = 0$ s, with $\omega = 2\pi$ and $\rho = 1\ \text{kg/m}^3$. To solve the associated governing equations, Dirichlet boundary conditions were applied at all boundaries for the velocity components together with zero gradient conditions for the pressure. Since all pressure boundary conditions are defined as Neumann conditions, the pressure was fixed at a point in the domain following Equation (8.5c).

Given that the solution was taken at $t = 0$ s the resulting velocity was used to determine the viscosity field using the Cross-WLF model, with the parameters given in Table 8.1. The solution for each of the primitive variables over the unit square domain, $(x, y) \in [0, 1] \times [0, 1]$ m, are depicted in Figure 8.1 together with the resulting viscosity field.

Table 8.1: Parameters of Cross-WLF model used in Taylor-Green test case.

Parameter	Value	Units
A_1	26	[-]
$\tilde{A}_2$	51.6	[K]
D_1	1×10^8	[Pa · s]
D_2	250	[K]
D_3	0	[K/Pa]
τ^*	10	[Pa]
n	0.2	[-]

Equations (8.1) and (8.2) were solved using the PIMPLE algorithm operating in SIMPLEC mode. The discretization schemes employed are summarized in Table 8.2. For the momentum balance equation, the symmetric Gauss-Seidel algorithm was used with a relative tolerance of 0.1. For the pressure equation, a preconditioned Conjugate Gradient algorithm with a Diagonal-based Incomplete Cholesky preconditioner was used, with a relative tolerance of 0.01. The iterative process was considered to be converged when the normalized initial residual of both pressure and velocity reached a value below 1×10^{-9}. An implicit under-relaxation factor of 0.99 was applied to the momentum balance equation.

Table 8.2: Finite volume discretization schemes for the Taylor–Green test case.

Operator / term	Discretization scheme	OpenFOAM terminology
Time derivative, $\partial\phi/\partial t$	Steady-state	`steadyState`
Gradient, $\nabla\phi$	Central differencing	`Gauss linear`
Convective term $\nabla \cdot (\rho\mathbf{uu})$	Linear upwind	`Gauss linearUpwind grad(U)`
Laplacian, $\nabla \cdot (\Gamma\nabla\phi)$	Central differencing	`Gauss linear orthogonal`
Interpolation from cell centers to faces	Linear interpolation	`linear`
Surface-normal gradient, $\nabla_f^{\perp}\phi$	Orthogonal	`orthogonal`

The results obtained for the order-of-convergence are summarized in Table 8.3. For the chosen solution and model settings, and within the considered grid resolution, all evaluated error norms for the velocity components show a slightly higher order-of-convergence than the theoretical second-order on the coarsest meshes. With progressive grid refinement, the error norms converge towards the theoretical second-order value, indicating a correct implementation of the Cross–WLF rheological model and the SIMPLEC methodology.

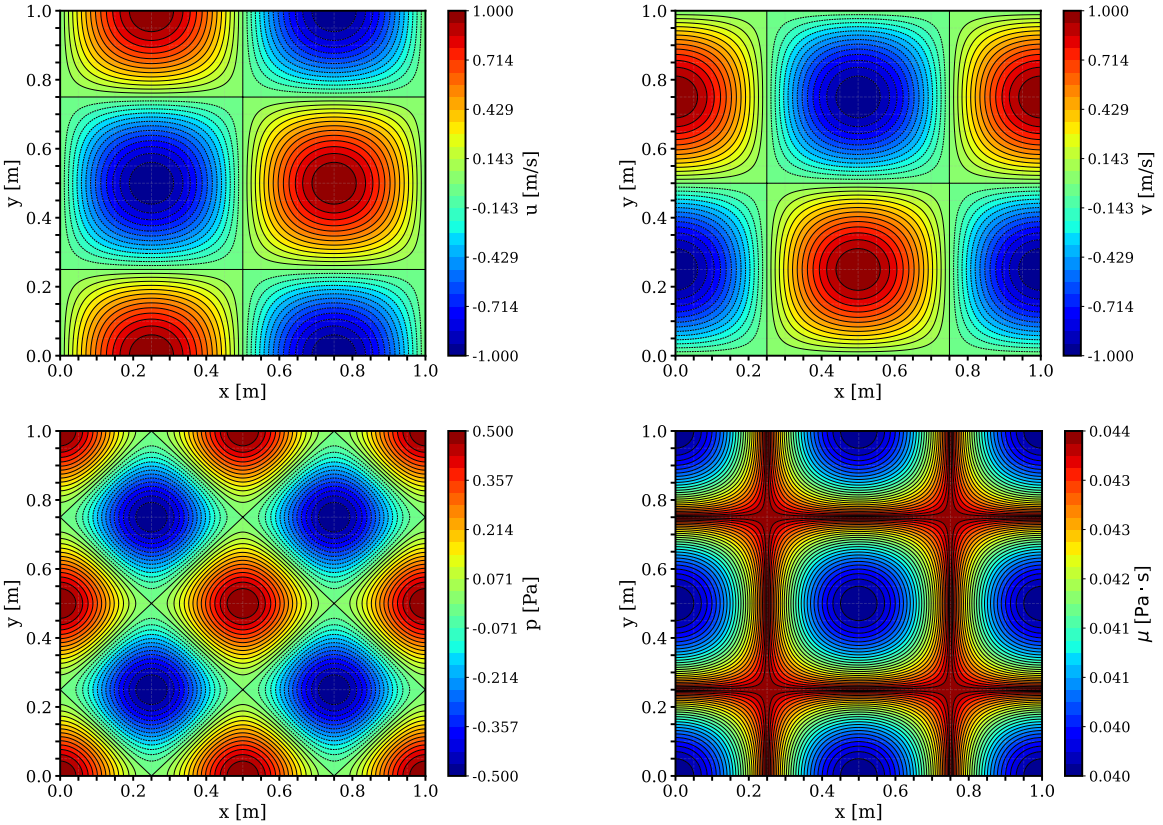


Figure 8.1: Solution field for u, v, p and resulting viscosity μ in Taylor-Green test case

Table 8.3: Convergence orders obtained for the different error norm and variables u, v, and p for the Taylor-Green test case with Cross-WLF rheological model.

Mesh	L^1			L^2			L^∞		
	u	v	p	u	v	p	u	v	p
Mesh 1 $\longrightarrow$ *Mesh 2*	2.41	2.48	1.44	2.36	2.38	1.47	1.91	1.87	1.51
Mesh 2 $\longrightarrow$ *Mesh 3*	2.14	2.14	1.83	2.10	2.11	1.84	1.93	1.92	1.86
Mesh 3 $\longrightarrow$ *Mesh 4*	2.04	2.03	1.95	2.02	2.03	1.95	1.96	1.96	1.96
Mesh 4 $\longrightarrow$ *Mesh 5*	2.00	1.99	1.99	2.00	1.99	1.99	1.98	1.98	1.99

8.3.1.2 Equation-of-state

To assess the correctness of the implementation of the equation-of-state, a two-dimensional solution following the work of Veeraragavan and coworkers [81] was used, reading:

$$u = 1 + \frac{1}{10}\cos\left(\frac{5\pi x}{3}\right) + \cos\left(\pi y\right) - \frac{1}{10}\cos\left(\frac{5\pi xy}{3}\right), \tag{8.6a}$$

$$v = \frac{9}{10} - \frac{9\sin\left(\frac{\pi y}{2}\right)}{10} - \frac{\cos\left(\frac{3\pi x}{2}\right)}{50} + \frac{\cos\left(\frac{3\pi xy}{2}\right)}{50}, \tag{8.6b}$$

$$T = 500 + 20\sin\left(\frac{3\pi y}{4}\right) - 5\cos\left(\frac{3\pi x}{2}\right), \tag{8.6c}$$

$$p = 100000 + 100\sin\left(\frac{3\pi x}{4}\right) + 100\cos\left(\pi y\right) + 100\cos\left(\frac{5\pi xy}{4}\right). \tag{8.6d}$$

The geometry and meshes employed for this case study are the same as those used previously, described in Section 8.3.1.1.

Figure 8.2 depicts the solution fields for the velocity components, temperature, and pressure over the unit square domain. In this test case, for the velocity, a Neumann boundary condition was applied to bottom boundary ($y = 0$ m) with corresponding Dirichlet condition for the pressure. All other boundaries use the opposite configuration, i.e., Neumann for velocity and Dirichlet for pressure. Dirichlet conditions were applied on all boundaries for the temperature field. Only the upper branch of the modified-Tait model was considered for verification purposes with the model parameters reported in Table 8.4.

The Navier-Stokes equations were solved using the SIMPLEC strategy, with discretization schemes as reported in Table 8.2 with the exception of the term $\nabla \cdot (\rho \mathbf{uu})$, which was discretized with the central difference scheme. The momentum balance and pressure equation were solved in a similar manner to the one reported in Section 8.3.1.1. Additionally, the temperature equation was solved through a Bi-Conjugate Gradient routine using the Diagonal-based Incomplete LU (DILU) preconditioner, with a relative tolerance of 0.01. The iterative algorithm was considered converged when the initial normalized residual of all primitive variables fell below 1×10^{-8}. An implicit under-relaxation of 0.98 and 0.99 were used for the momentum and energy equations, respectively.

The results for order-of-convergence with progressive grid refinement are reported in Table 8.5. Observation of the results shows that second-order convergence is achieved for all observed error norms for the velocity components and temperature. For the pressure, the L^1 norm appears to move towards second-order with the L^2 and L^∞ approaching first-order. This behavior is consistent with the adopted pressure formulation and boundary treatment [24]. Overall, these results support a correct implementation of the compressible portion of the solver.

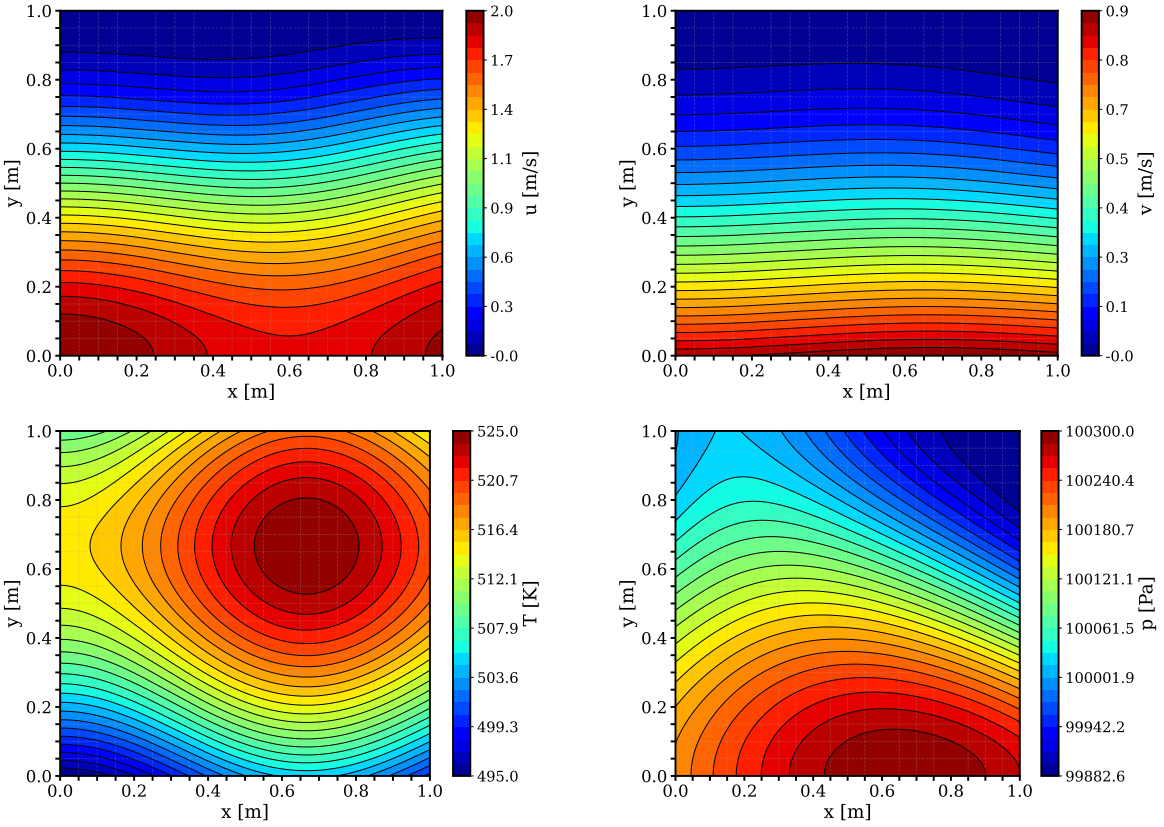


Figure 8.2: Solution field for u, v, T and p for the Veeraragavan et al. [81] solution

Table 8.4: Parameters of the modified-Tait model. All omited parameters are set to zero.

Parameter	Value	Units
b_{2L}	3×10^{-3}	$[\mathrm{m}^3\,\mathrm{kg}^{-1}\mathrm{K}^{-1}]$
b_{3L}	1×10^{4}	[Pa]
b_5	373	[K]
C	0.0894	[-]

Table 8.5: Convergence orders obtained for the different error norms and variables u, v, p and T test case with modified-Tait model.

	L^1				L^2				L^∞			
Mesh	u	v	T	p	u	v	T	p	u	v	T	p
Mesh 1 ⟶ Mesh 2	1.96	1.99	2.00	1.97	1.97	1.99	2.00	1.75	1.95	1.98	2.00	0.65
Mesh 2 ⟶ Mesh 3	1.98	2.00	2.00	1.98	1.98	2.00	2.00	1.69	1.99	1.99	2.00	0.85
Mesh 3 ⟶ Mesh 4	1.99	2.00	2.00	1.99	1.99	2.00	2.00	1.63	2.00	1.99	2.00	0.94
Mesh 4 ⟶ Mesh 5	1.99	2.00	2.00	1.98	1.99	2.00	2.00	1.58	2.00	2.00	2.00	0.97

8.4 Phase advection

To assess the performance of the different phase advection schemes, we employ a test case that produces severe deformations on the VoF indicator field. The problem consists of stretching and thinning a two-dimensional droplet in a vortical flow [16, 51]. In this case study, a droplet of radius 0.15 m is placed inside a unit square domain $(x, z) \in [0, 1] \times [0, 1]$ m at $(x, z) = (0.5, 0.75)$ with a velocity field imposed at every time step by:

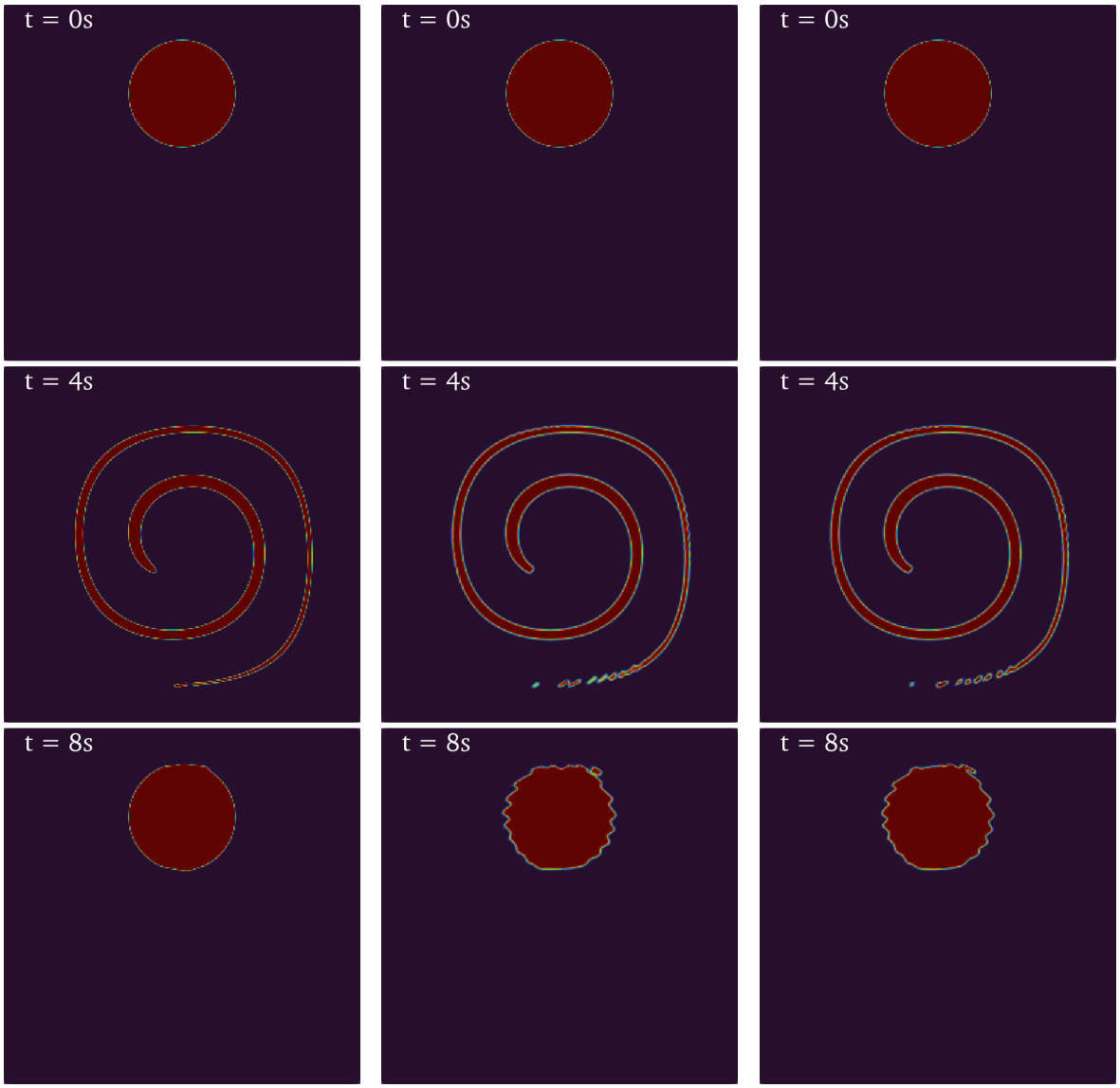
$$\mathbf{u}\left(x, z, t\right) = \cos\left(\frac{\pi t}{8}\right) \begin{bmatrix} -\sin^2\left(\pi x\right) \sin\left(2\pi z\right) \\ \sin^2\left(\pi z\right) \sin\left(2\pi x\right) \end{bmatrix}. \tag{8.1}$$

This solenoidal flow produces a spiraling deformation with maximum stretch at $t = 4$ s. For $4s < t \leq 8s$, the flow reverses and the deformation unwinds, so the VoF field should ideally revert to its original shape. The test is performed on an orthogonal structured mesh with 256 uniform cells, in each Cartesian direction. Three VoF advection algorithms are compared using a constant time step of 5×10^{-4} s: an explicit algebraic VoF method using the MULES algorithm [51], an implicit algebraic VoF solution of Equation (8.4) using the interface compression term defined in [51], and the explicit geometrical VoF (IsoAdvector [16]) algorithm. For both algebraic variants, the convective term involving the VoF field is discretized using the `vanLeer` scheme and the interface compression term with a `interfaceCompression` scheme. For the MULES algorithm, two correction steps of the VoF field were performed per time-step without sub-cycling. For the IsoAdvector algorithm, a tolerance of 1×10^{-8} was used to divide the cells into sub-cells and three bounding steps were applied to enforce boundedness of the VoF field.

Figure 8.3 compares the results obtained from all algorithms at different times. At $t = 4$ s, the time of maximum stretch, all advection schemes show a similar overall shape with some degree of pinching at the tail of the spiral. This effect being more pronounced on the algebraic variants. Additionally, differences can be observed at the head of the spiral, where more rounded features are predicted with the algebraic methods. After the flow reversal, the IsoAdvector algorithm outperforms its algebraic counterparts, which recover a markedly distorted version of the original shape.

For this test case, the quality of the solution was assessed by computing the L^1-error norm between the initial ($t = 0$ s) and final ($t = 8$ s) configurations, together with the required computational time. The IsoAdvector algorithm yielded an error of $L^1 = 0.0005$ and a computational time of 934 s, the MULES routine gave an $L^1 = 0.0041$ with 636 s and the implicit algebraic approach $L^1 = 0.0036$ with 529 s. The simulations were executed in single-core on a workstation with a AMD Ryzen Threadripper 3970X 32-Core processor, with 128 Gb of RAM (2666 MHz) and running Ubuntu 22.04.

For the considered conditions, the geometric VoF method shows an L^1-error that is almost one order magnitude smaller than the algebraic counterparts, however its computational cost is $\approx 77\%$ higher than the fastest algebraic method. This trade-off between improved interface accuracy and



(a) isoAdvector (b) MULES (c) implicit

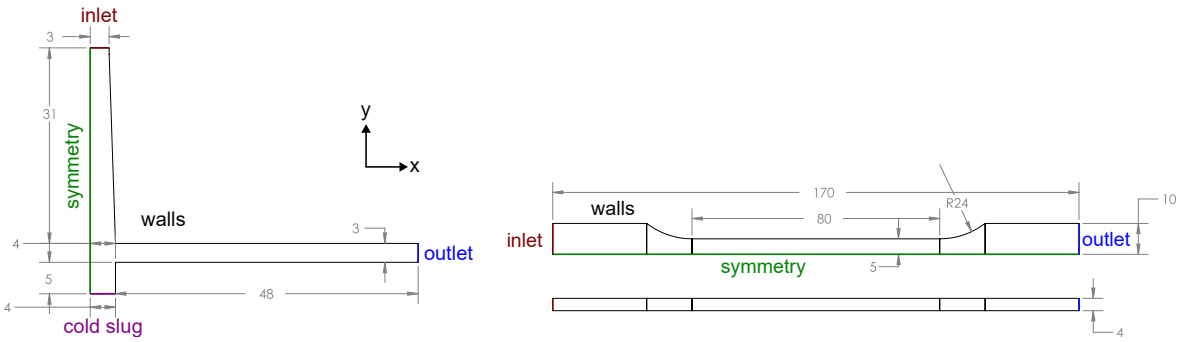
Figure 8.3: Comparison of algebraic and geometric VoF algorithms in vortical flow.

computational time is consistent with the literature on VoF algorithms [16]. It is worth noting that the implemented geometric algorithm has also been reported to be faster than MULES for certain test cases [82], indicating that the relative performance can be case dependent.

For injection-molding applications, the IsoAdvector algorithm is therefore attractive if a more accurate representation of the interface is required and the additional cost is acceptable. Nevertheless, the current implementation of geometric algorithm still exhibits some known drawbacks, with limitations in grid convergence and generation of spurious currents when compared with the MULES algorithm [18].

8.5 Analysis of dynamic outlet conditions

In this section, two tests are carried out to investigate the behavior of the dynamic outlet boundary conditions with respect to the filling pattern, volume/(mass) balance, and inlet pressure evolution. The first test uses a two-dimensional cross-section of a center-gated disk (CGD), to study a 2D planar flow. The second, a three-dimensional dogbone laboratory specimen. Both geometries are depicted in Figure 8.4, together with the corresponding boundary names.



(a) CGD cross-section

(b) dogbone specimen

Figure 8.4: Geometry and boundary definition for specimens used in the dynamic outlet analysis (dimensions in mm)

8.5.1 Cross-section of center-gated disk

The filling analysis of the cross-section of the center-gated disk was carried out assuming an incompressible flow with Newtonian viscosity of $100\,\mathrm{Pa{\cdot}s}$ and a density of $1000\,\mathrm{kg/m^3}$. The properties of the pseudo-air were defined as a ratio of 10^{-3} to those of the polymer. The computational domain was generated using `gmsh` [83] and contained $7\,773$ cells, with 15 cells across the disk thickness. The grid was generated to maintain a quasi-uniform cell size across the domain, with a maximum aspect-ratio, of 1.27, a maximum non-orthogonality of $1.85°$, and a maximum skewness of 0.037.

Four sets of boundary conditions were tested. Their definitions, together with the used

discretization schemes and solution algorithms, are reported in Tables 8.6, 8.7 and 8.8, respectively. The VoF equation was solved with Explicit MULES, using three correction steps. The momentum balance equation was solved with a Biconjugate Gradients Stabilized Method (BICGStab) using DILU as a preconditioner. The pressure equation was solved with a Geometric Algebraic Multigrid (GAMG) solver using DIC as a smoother.

Table 8.6: Combination of boundary conditions used in the CGD cross-section tests. The acronyms *zG*, *fV*, *dO*, and *ns* denote: null normal gradient, fixed value, dynamic outlet, and no-slip, respectively. For the dynamic outlets, the bracketed label indicates the weighting function used (jump, sigmoidal, or linear)

	BC1			BC2			BC3			BC4		
	p	$\mathbf{u}$	α	p	$\mathbf{u}$	α	p	$\mathbf{u}$	α	p	$\mathbf{u}$	α
inlet	*zG*	*fV*	*fV*	*zG*	*fV*	*fV*	*zG*	*fV*	*fV*	*zG*	*fV*	*fV*
walls	*zG*	*ns*	*zG*	*dO* [jump]		*zG*	*dO* [sigmoidal]		*zG*	*dO* [linear]		*zG*
cold slug	*zG*	*ns*										
outlet	*fV*	*zG*										
symmetry	symmetryPlane											

The VoF field was initialized with $\alpha = 1$ in the layer of cells adjacent to the inlet, corresponding to a global polymer volume fraction of approximately 0.002.

The inlet velocity was prescribed to obtain a filling time of 0.1 s, which corresponds to a uniform velocity of $946.6(6)$ mm/s along the negative y-direction. The governing equations were solved to reach 0.1 s, or until the global VoF indicator reached 0.99.

Within each time-step, convergence was considered to be achieved when the initial normalized residuals of the momentum balance and pressure equations fell bellow 1×10^{-4} and 1×10^{-5}, respectively. In addition, the relative change in the area-weighted inlet pressure between consecutive outer iterations, Equation (8.1), was used as a quantity of interest to support the convergence assessment, using a tolerance of 1×10^{-4}.

$$\Delta p = \frac{\left| p^{k-1}_{\text{inlet}} - p^{k}_{\text{inlet}} \right|}{\left| p^{k}_{\text{inlet}} \right|} \tag{8.1}$$

$$p_{\text{inlet}} = \frac{\sum_{f \in \text{inlet}} p_f A_f}{\sum_{f \in \text{inlet}} A_f} \tag{8.2}$$

To control the convergence, the solver was allowed to perform up to 150 outer iterations per time-step, with 5 PISO correctors. An implicit under-relaxation factor of 0.9 was applied to the

momentum balance equation in all outer iterations except the final one, where a value of 1.0 was used.

Table 8.7: Discretization schemes for the CGD cross-section.

Operator / term	Discretization scheme	OpenFOAM terminology
Time derivative, $\partial\phi/\partial t$	Implicit Euler	`Euler`
Gradient, $\nabla\phi$	Least-squares	`leastSquares`
Convective terms:		
$\nabla \cdot (\rho\mathbf{u}\mathbf{u})$	Linear upwind	`Gauss linearUpwindV grad(U)`
$\nabla \cdot (\alpha\mathbf{u})$	Van Leer	`Gauss vanLeer`
$\nabla \cdot (\phi_r\alpha\,(1-\alpha))$	Interface compression	`Gauss interfaceCompression`
$\nabla \cdot \left(\mu\left[(\nabla\mathbf{u})^{\mathrm{T}} - \frac{2}{3}\mathbf{I}\,(\nabla\cdot\mathbf{u})\right]\right)$	Central differencing	`Gauss linear`
Laplacian, $\nabla \cdot (\Gamma\nabla\phi)$	Central differencing with non-orthogonality correction	`Gauss linear corrected`
Interpolation from cell centers to face centers	Linear interpolation	`linear`
Surface-normal gradient, $\nabla_f^{\perp}\phi$	Corrected surface-normal gradient	`corrected`

Table 8.8: Solution settings for the CGD cross-section.

α		p		$\mathbf{u}$	
nAlphaCorr	3	solver	GAMG	solver	BiCGStab
nAlphaSubCycles	1	smoother	DIC	smoother	DILU
cAlpha	1	tolerance	1×10^{-9}	tolerance	1×10^{-11}
		relTol	0.01	relTol	0

The filling patterns as a function of time for all configurations considered are shown in Figure 8.5. The temporal sequences correspond to snapshots taken every 0.005 s of simulation time, showing the clipped sub-region where $\alpha \geq 0.5$, a value typically used to define the position of the interface [21].

Analysis of the figure, shows that imposing a no-slip boundary condition at the walls (BC1) limits the displacement of the pseudo-air. This is specially pronounced in the cold-slug region. As a consequence, the secondary-phase remains trapped, with the interface advancement stalling and leading to an under-filled part. This illustrates the overly restrictive nature of the no-slip condition when using incompressible settings for IM simulations. With compressibility available on the secondary phase, this limitation is expected to be alleviated, however, there is no guarantee that this will result in a fully filled part. This highlights the need for a more suitable numerical strategy to enable adequate advancement of the interface and effective displacement of the secondary-phase thereby improving the filling dynamics of the part.

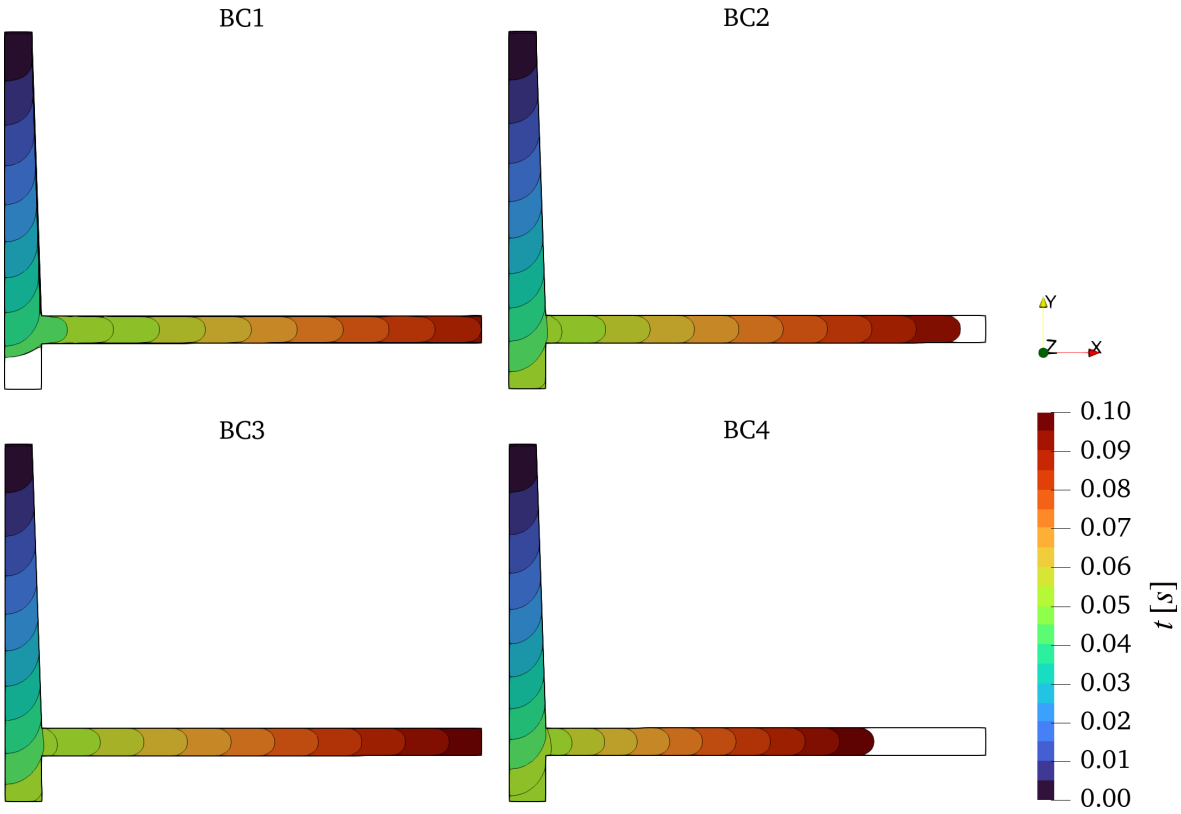


Figure 8.5: Comparison of VoF field for the cross-sectional disk as a function of time. Sequence was obtained by clipping the VoF field with $\alpha \geq 0.5$

The dynamic outlet conditions (BC2-BC4) yield a more realistic filling pattern, as they allow phase transport through the boundaries. However, the dynamic pressure-velocity pairs are implemented as a phase-weighted mixed boundary condition, where the pressure treatment transitions between two limiting states via a weighting factor that is dependent on α. Although an attractive solution, two significant caveats must be considered:

- Loss of pressure reference when the dynamic outlet becomes partially filled with polymer. As filling progresses, the pressure boundary might no longer provide a pressure reference. Consequently, the system behaves like a closed incompressible domain, with the pressure being determined up to an additive constant. Without an explicit pressure reference, the system becomes undetermined,leading to solver failure. This issue is observed when using BC2, which terminated prematurely at $\alpha \approx 0.98$. A straightforward solution would be to enforce a pressure reference, however, for the purpose of this work, such a solution was not pursued.

- Linking the pressure-velocity boundary conditions to the VoF variable can lead to numerical chatter, which was observed in the inlet pressure history under the dynamic outlet conditions, with particularly pronounced effects for BC4. This issue can be mitigated by using a smaller time-step/ tightening the tolerances of the area-weighted inlet pressure. Additionally, the pressure oscillations can also be reduced by adopting a α-lagged strategy, see Appendix B

within each time step: the pressure–velocity system is iterated with α held fixed, and the VoF advection equation is solved after the pressure–velocity system has converged. This strategy as already been used in the context of the SIMPLE algorithm in IM [21].

Figure 8.6 compares the pressure histories obtained with the different boundary conditions using: (i) the standard solution strategy, where α and the mixture properties are updated every outer iteration, and (ii) the α-lagged strategy. To reduce excessive chatter, the former was executed with 4 PISO correctors and a Courant number and interface Courant number of 0.01, whereas the latter used a value of 0.1.

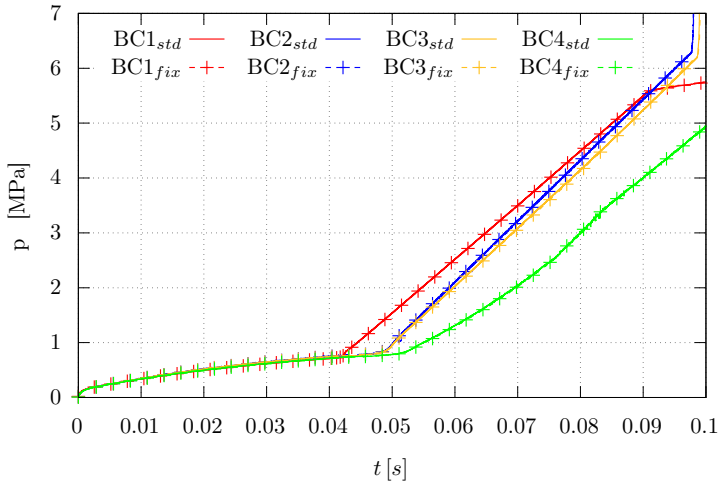


Figure 8.6: Comparison of pressure history at the inlet between standard approach, $(\cdot)_{std}$, and α-lagged strategy, $(\cdot)_{fix}$

Analysis of the figure shows that the results of the α-lagged strategy match fairly well with the standard approach, while allowing a higher Courant number and lower computational time. A similar trend is also observed for the computed velocity profiles, which will be presented latter in this section. Despite the observed numerical chatter in the standard approach, we find that useful engineering results can still be obtained with these boundaries. However, these findings highlight that further additional work is required to improve the stability of the boundaries in incompressible settings.

The sigmoidal (BC3) and linear (BC4) conditions do not enforce a traction-free condition, however, both are able to complete the simulation without triggering the null-space issue observed in BC2. The two conditions behave differently with respect to volume conservation. The linear variant exhibits a substantial loss of polymer, whereas the sigmoidal is able to more adequately preserve the injected volume within the computational domain.

To further quantify the imbalance associated with the dynamic outlets, the following metrics were used:

$$\tilde{\phi}_f = \frac{(\alpha\phi)_f \, \Delta t}{\sum V_c}, \tag{8.3a}$$

$$m_{\text{loss}}(T) = \frac{\sum_{b\neq\text{inlet}} \sum_{t\leq T} \sum_{f\in b} \left(\tilde{\phi}_f\right)_+}{\sum_{t\leq T} \sum_{f\in\text{inlet}} \left|\left(\tilde{\phi}_f\right)_-\right|}, \tag{8.3b}$$

$$m_{\text{gain}}(T) = \frac{\sum_{b\neq\text{inlet}} \sum_{t\leq T} \sum_{f\in b} \left|\left(\tilde{\phi}_f\right)_-\right|}{\sum_{t\leq T} \sum_{f\in\text{inlet}} \left|\left(\tilde{\phi}_f\right)_-\right|}, \tag{8.3c}$$

where the subscript b refers to a boundary patch, T the total simulation time and subscripts "+" and "−" the outward and inward flux contributions. This weighted volumetric flux was evaluated at each time step of the calculation process.

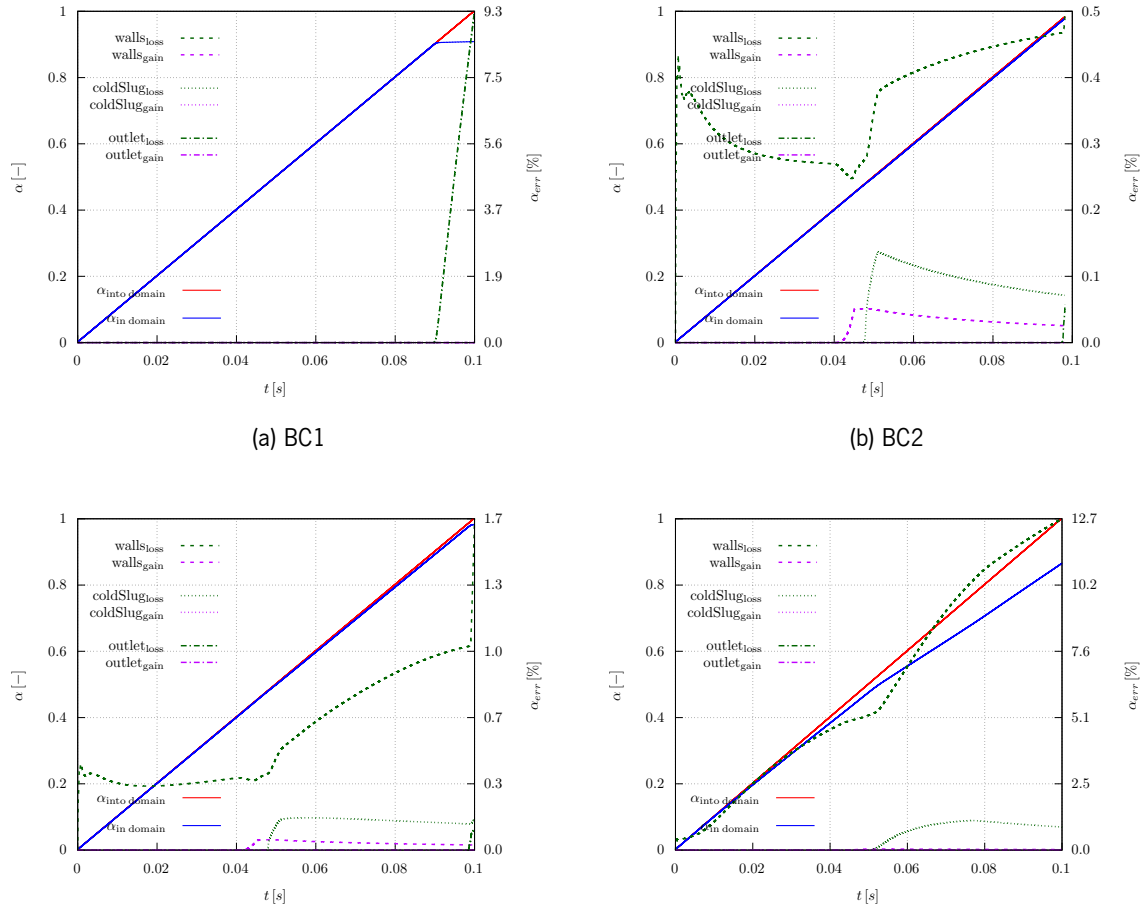
Figure 8.7 summarizes the volume conservation achieved with the different boundary conditions. For BC1, no imbalance is observed until the polymer phase reaches the outlet, as the impermeability from the no-slip condition prevents flux through the boundary. Once the interface reaches the outlet patch, the polymer is allowed to flow out of the computational domain and the observed imbalance is a direct consequence of the polymer flux through the outlet.

For the dynamic outlet conditions, an imbalance is observed during the flow history, consistent with the qualitative results in Figure 8.5. The dynamic character of these boundaries allows the transport of α across the boundary (polymer leakage), but also admits inflow when backflow occurs. Analysis of Figure 8.7 shows that BC4 exhibits the largest α-imbalance, followed by BC3 with less than 2% loss and finally BC2 with a value lower than 0.5%.

The filling pattern in Figure 8.5 was generated with the conventional threshold $\alpha \geq 0.5$, however, applying a stricter criterion, $\alpha \geq 0.9$, reveals additional insight into the filling dynamics. As shown in Figure 8.8, near the end of filling, the wall-adjacent cells remain only partially filled. This shows that, although the flow front has reached the wall region according to the $\alpha \geq 0.5$ criterion, the local mixture composition has not evolved towards a pure polymer state. This observation shows that the choice of flow front criteria can bias the interpretation of the results. In the VoF formulation, the material properties are given by a mixture rule that uses α as a weighting coefficient. Therefore, cells that still contain the interface lead to erroneous values of quantities of interest such as, viscosity, pressure drop and wall shear-stress, if they are interpreted as fully filled with polymer.

Among the considered cases, only BC4 produces a satisfactory near-wall filling $\alpha \geq 0.9$. This improvement is accompanied by the largest volume loss, indicating a trade-off between enhanced local near-wall filling and reduced phase conservation due to outflow across the dynamic outlet.

Building on the observation that the flow front criteria can mask partially filled cells, we use the



(a) BC1

(b) BC2

(c) BC3

(d) BC4

Figure 8.7: VoF balance of CGD cross-section test case

analytical solution for plane Poiseuille flow between two stationary plates, Equation (8.4), to quantify the deviations due to under-filled cells near the walls. For this purpose, after filling the sprue region, and once the flow has progressed sufficiently with the pseudo-air phase allowed to leave the computational domain, the solution is expected to approach the corresponding single-phase laminar profile at the middle of the disk section, $x = 28$ mm, which reads as follows:

$$u(y) = \frac{3}{2} u_{\mathrm{av}} \left(1 - \left(\frac{y}{H}\right)^2\right) \quad \text{and} \quad v, w = 0, \tag{8.4}$$

where u_{av} denotes the cross-sectional average velocity, y is the wall normal coordinate measured from the mid-plane of the channel, and H is the channel half-thickness. For this case, the average velocity was computed from the inlet flux (Q_{inlet}) as $u_{\mathrm{av}} = |Q_{inlet}|/A_{cross}$, with A_{cross} being the area of the disk cross-section.

The inlet pressure history, taken at every time-step, and the corresponding velocity profile near the end of filling ($t = 0.09$ s) are shown in Figure 8.6 and Figure 8.9, respectively.

All cases exhibit a similar initial pressure evolution while the sprue region is being filled.

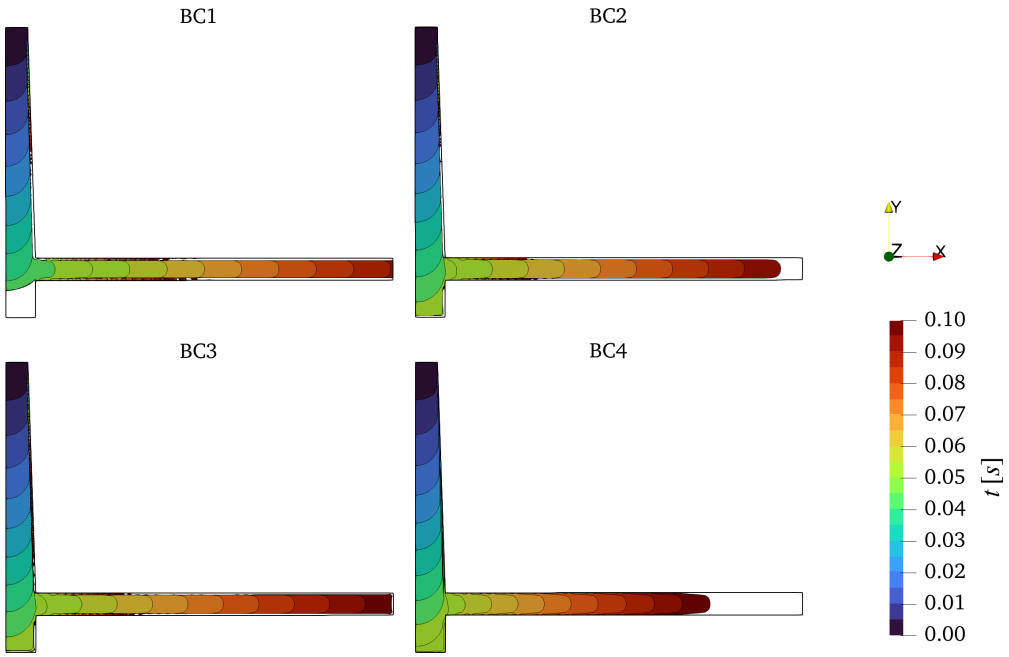


Figure 8.8: Comparison of VoF field for the cross-sectional disk as a function of time. Sequence was obtained by clipping the VoF field with $\alpha \geq 0.9$

Afterwards, because BC1 underfills the cold-slug region, it diverts the flow into the disk section earlier than the other configurations. As a result, the inlet pressure begins to rise sooner. At later times, once the flow-front reaches the outlet, the pressure rise becomes less steep, as observed by the reduction in slope on the pressure curve. Analysis of BC2 and BC3 shows that the pressure evolution is broadly similar. The sigmoidal condition allows cells to be locally filled to a higher degree, thus having a slightly higher pressure slope. Finally, the linear condition, shows the lowest pressure values consistent with the large loss of polymer through the boundaries.

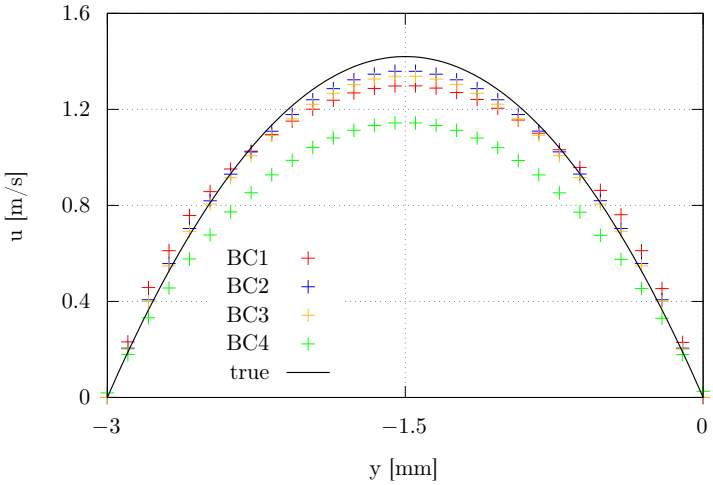


Figure 8.9: Velocity profile at $x = 28$ mm at 0.09 s

The measured velocity profiles on Figure 8.9 are all parabolic, independent of the boundary condition used, which is to be expected given the similarity to laminar pressure-driven flow between

parallel plates.

Among the tested cases, BC2 shows the closest agreement with the analytical profile, followed closely by BC3 and BC1. Finally, BC4 exhibits the largest deviation from the analytical solution. The discrepancies in BC1-BC3 are attributed to incomplete local filling of the cells at the sampling location. Due to the under-filling of the cells adjacent to the wall, they contribute to the pressure-velocity system with α-weighted properties. As a result, the viscosity near the walls differs from the single-phase value assumed by the analytical solution, leading to measurable deviations in the predicted velocity profile.

The larger mismatch observed for BC4 is attributed to the under-filling of the disk section. At the end of the simulation, the flow at the sampling location has not yet reached a fully developed regime, and the velocity profile is still evolving.

When using the VoF methodology for modeling the IM filling stage, non-inlet boundaries are commonly assigned with a null normal gradient condition for the phase-fraction [15, 19]. While this choice is often adequate for generic free-surface flows, our results show that, when using dynamic outlets, the use of null normal gradient allows non-physical phase exchange at the dynamic boundaries, leading to a mass imbalance where polymer is either lost from, or injected into, the computational domain.

To address this issue, an alternative treatment was envisaged and tested, mimicking a semi-permeable membrane, in line with [12]. Rather than assigning $\nabla\alpha \cdot \mathbf{n} = 0$, the α-flux across the dynamic walls is set to zero, $(\alpha\mathbf{u})_f \cdot \mathbf{n} = 0$. Conceptually, in the proposed approach the displaced pseudo-air is allowed to flow out of the computational domain, but the polymer is retained.

For the following analysis the previously established nomenclature (BC2–BC4) for the boundary types is kept, since only the α boundary condition has been modified to enforce zero polymer flux across the dynamic outlets. The solution algorithm employs the α-lagged strategy within each time step, while all other settings remain unchanged. Since BC1 does not benefit from this approach, it was kept for comparison purposes.

The resulting filling progression is shown in Figure 8.10, using a threshold value $\alpha \geq 0.9$ to highlight under-filled regions. Analysis of the figure shows that BC2 and BC3 exhibit similar under-filled regions using both null normal gradient and null-flux treatments. The most significant improvement is obtained for BC4, where no appreciable under-filling is observed. This improved filling behavior is also reflected in the inlet pressure history and the velocity profile depicted in Figure 8.11. For the selected settings, the dynamic outlets display an overall similar pressure evolution, with BC4 exhibiting a slightly steeper pressure increase, consistent with its higher local degree of filling. In addition, BC4 now provides the closest match to the analytical Poiseuille flow solution, indicating that preventing polymer leakage while enabling adequate near-wall advancement substantially improves the approach to reach the expected single-phase, fully developed regime.

It is important to emphasize that this approach still carries important caveats. The tests performed allowed to conclude that imposing a hard constraint on the phase-flux at the boundary

needs to be handled carefully. Moreover, as mentioned before, the dynamic outlets require tighter tolerances to allow a stable pressure history and avoid excessive numerical chatter. Monitoring the relative change in area-averaged inlet pressure between successive outer iterations provided a practical and sensitive metric for controlling this phenomenon. When the convergence criteria were relaxed, the resulting velocity profiles remained qualitatively reasonable. However, the pressure history exhibited pronounced numerical chatter, underscoring the need of further developments to improve the robustness of the approach.

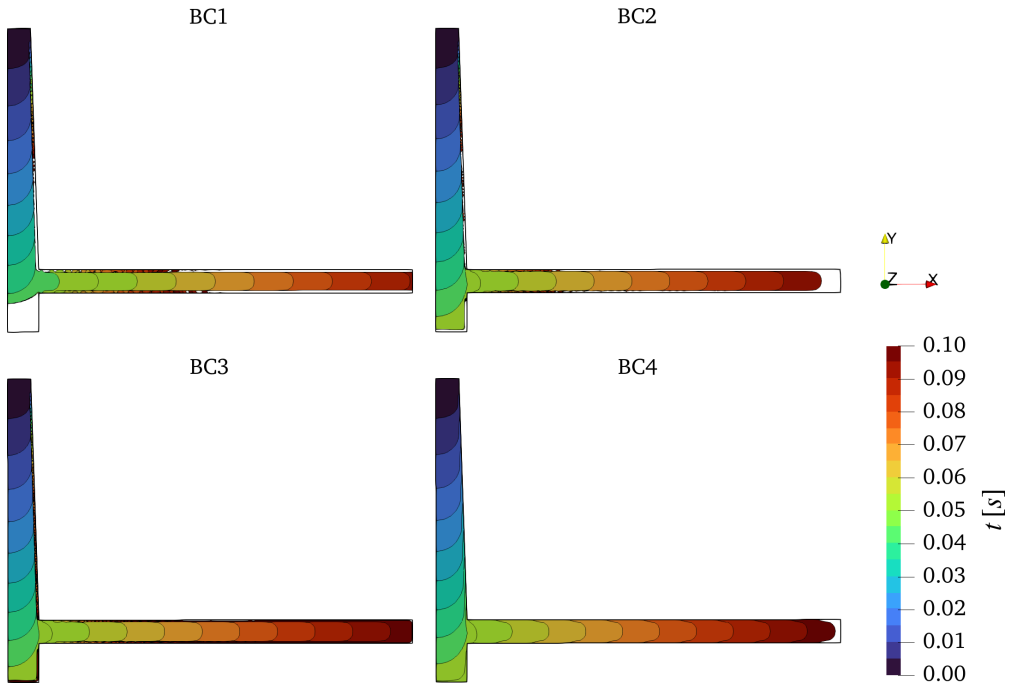
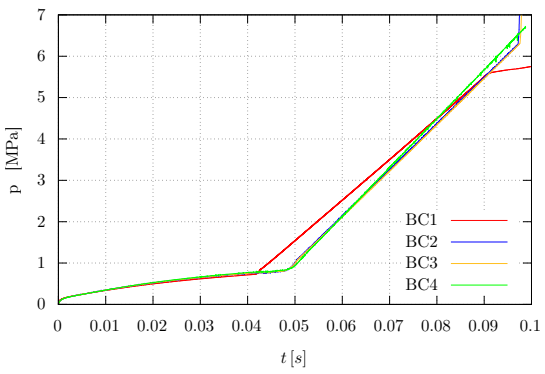
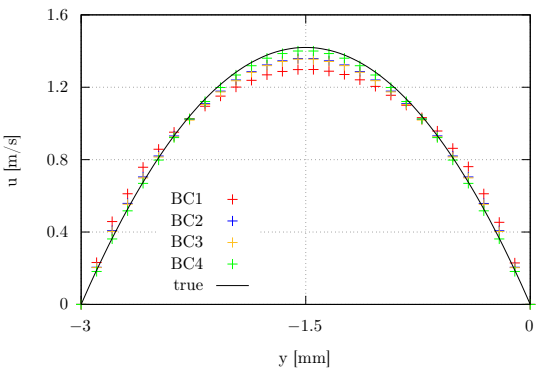


Figure 8.10: Comparison of VoF field for the CGD cross-section as a function of time. Conditions BC2-BC4 have null-flux through the dynamic outlets. The image sequence was obtained by clipping the VoF field with $\alpha \geq 0.9$



(a) Filling pressure



(b) Velocity profile at $x = 28$ mm at 0.09 s

Figure 8.11: Pressure history and velocity profile of the analyses conditions. Cases BC2-BC4 used a no α-flux boundary condition

8.5.2 Dogbone specimen

A tensile specimen, following ISO 527, was used as a three-dimensional benchmark. Since the specimen has symmetry along its width, only half of the geometry was considered. The computational model was generated with `gmsh` [83] and comprised $102\,300$ cells, with 11 along the thickness. The resulting mesh had a maximum non-orthogonality of 36° and a maximum skewness of 0.76.

The modified solution algorithm is summarized in Table 8.9. The transport equation of α was solved with the explicit MULES algorithm by sub-cycling the VoF equation three times and performing three correction steps per sub-cycle. The momentum balance equation was solved with a BICGStab using DILU as a preconditioner. The discretization schemes were the same used in Section 8.5.1, with the term $\nabla \cdot (\rho \mathbf{u}\mathbf{u})$ being discretized with linear upwind scheme.

Table 8.9: Modified solution settings for the dogbone test case.

α		$\mathbf{u}$	
nAlphaCorr	3	solver	BiCGStab
nAlphaSubCycles	3	smoother	DILU
cAlpha	1	tolerance	1×10^{-10}
		relTol	0

The material properties are the same as those used in Section 8.5.1. The associated incompressible flow governing equations were solved until reaching 1 s, or until the global polymer volume fraction reached 0.99, using a maximum Courant and interface Courant numbers of 0.5. The VoF field was initialized with $\alpha = 1$ in the layer of cells adjacent to the inlet, corresponding to a global polymer volume fraction of approximately 0.003. The inlet velocity was prescribed as a uniform value of 0.12 m/s along the positive x-direction. Within each time step, convergence was considered to be achieved when the initial normalized residuals of the momentum balance and pressure equations fell below 1×10^{-3} and 1×10^{-6}, respectively. In addition, a tolerance of 1×10^{-4} was defined for the relative change in the area-weighted inlet pressure between consecutive outer iterations. The system was solved using the α-lagged strategy. Similarly to CGD cross-section case study, an implicit under-relaxation factor of 0.9 was applied to the outer iterations of the momentum balance equation, with the exception of the last iteration, where a value of 1 was used. To meet these criteria, the solver was allowed up to 75 outer iterations per time-step, with 4 PISO correctors. The experiments used in this Section employ the same boundary conditions and nomenclature as those in Section 8.5.1 .

The filling patterns as a function of time for all configurations considered are shown in Figure 8.12 for a null-normal-gradient condition for α. The temporal sequences correspond to snapshots taken every 0.05 s of simulation time.

When using BC1, no-slip condition at the walls, a markedly irregular filling pattern is created. Several wave-like patterns develop throughout the filling of the cavity, indicating that the polymer is entrapping the pseudo-air along the walls, which is an unintended artifact when modeling general

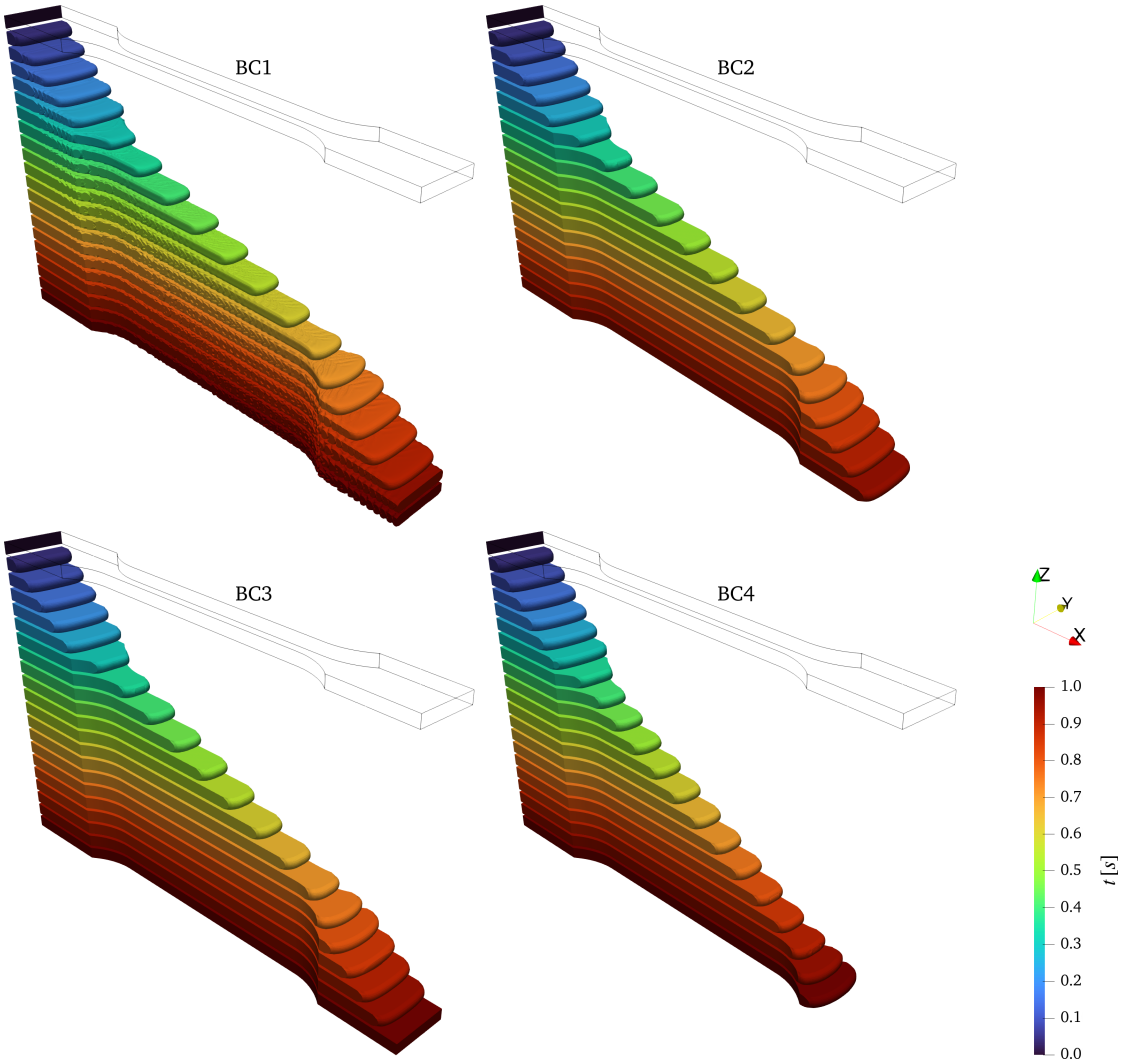


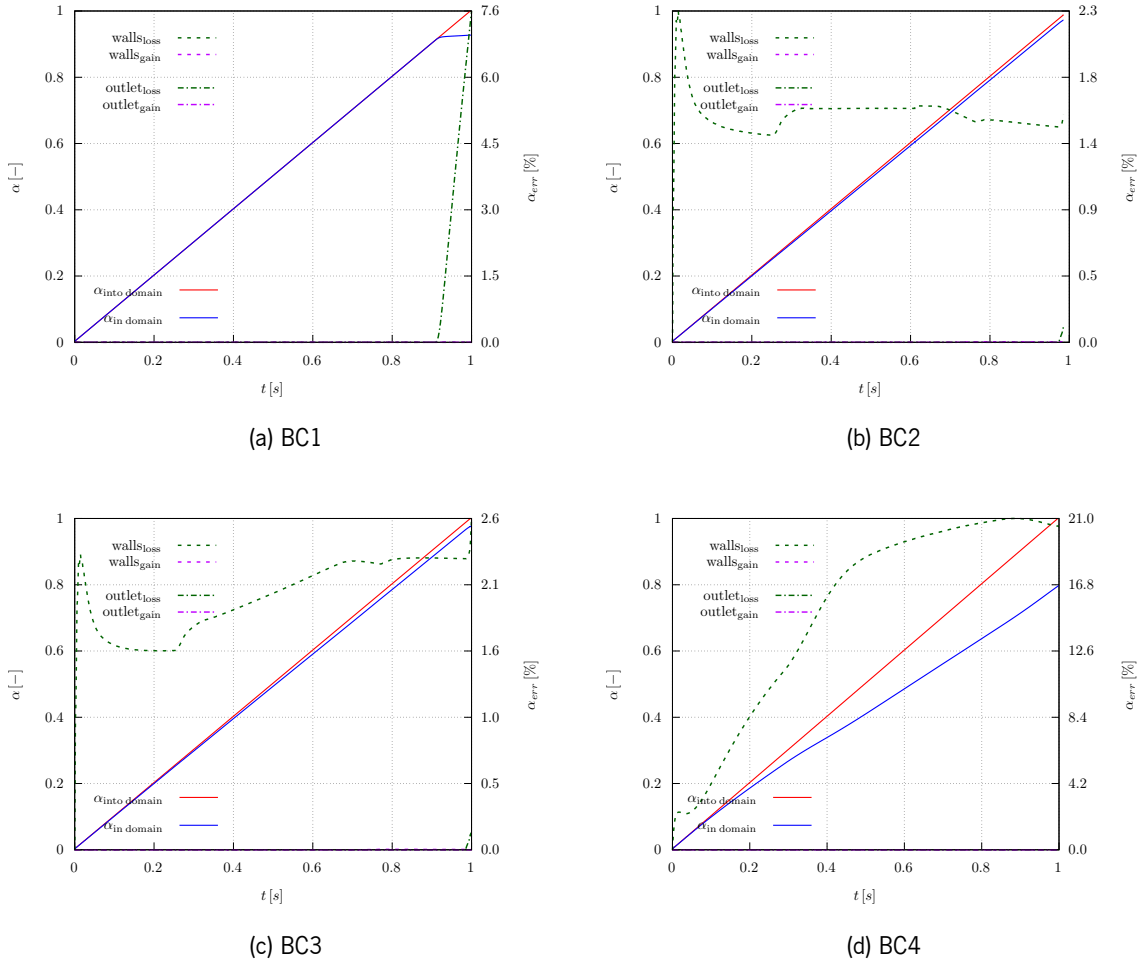
Figure 8.12: Comparison of the polymer region evolution for the dogbone case study. The sequence was obtained by clipping the VoF field with $\alpha \geq 0.5$

purpose IM. This pattern has been previously shown in the literature [14] for a different geometry, with the authors also attributing this phenomenon to entrapped air. This is in line with the behavior observed in Section 8.5.1 and in the literature [20], where the no-slip condition should only be applied to the polymer phase.

When using dynamic outlets, the trends are similar to the ones obtained in Section 8.5.1. The filling patterns are smoother then the ones obtained for BC1, for the clipping with $\alpha \geq 0.5$. Condition BC2 terminates prematurely due to the lack of a pressure reference, at $\alpha \approx 0.97$, whilst the other two conditions are able to finish the run.

The VoF imbalance is summarized in Figure 8.13. Conditions BC2 and BC3 behave relatively similarly, with less than 3% volume loss at the end of the simulation time, whilst BC4 loses approximately 21%. Similarly, BC4 is the condition with the highest local cell filling. Although no pronounced polymer mass is gained through the boundaries due to recirculation, attempts were made to clip backflow by introducing a switch that filters the Robin boundary condition, setting a

no-slip condition for velocity and a null pressure gradient upon detection of backflow. However, this approach introduced substantial difficulties in converging the linear system when using the standard solution approach, consequently this approach was not further considered.



(a) BC1

(b) BC2

(c) BC3

(d) BC4

Figure 8.13: VoF balance of dogbone test case

The pressure history for the considered boundary conditions is depicted in Figure 8.14. Three regions are identifiable in the figure. The first originates from the pressure increase required to fill the shoulder area near the gate, the second is characterized by a steeper pressure increase to fill the parallel section of the specimen, and the last is associated with a reduced rate of pressure increase to fill the shoulder area far from the gate. Conditions BC1-BC3 exhibit a similar pressure evolution during the filling of the first shoulder area, while BC4, due to its high mass loss, fills this region later. The figure also shows numerical chatter in the pressure history for conditions BC2 and BC4, with a more pronounced effect over a localized region for BC2. During the simulation, the solution of the α-equation became slightly unbounded, with $\alpha > 1$ in a limited number of cells. Since physical admissibility of this variable requires $0 \leq \alpha \leq 1$, a boundedness correction was enforced by clipping α to its admissible interval. This non-conservative correction was monitored and quantified as a mass imbalance, which remains consistent with the imbalance obtained when using dynamic outlet conditions.

The same set of simulations, using no-flux boundary conditions for α, is depicted in Figure 8.15.

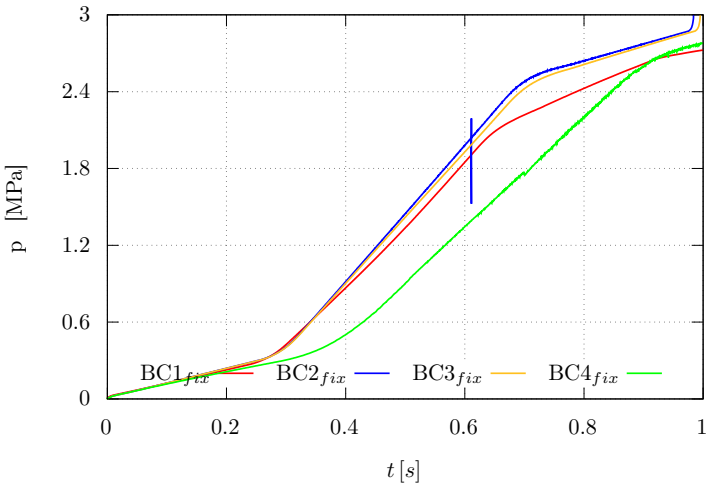


Figure 8.14: Pressure history for BC1-BC4 using the null-normal-gradient boundary conditions

Using the criterion $\alpha \geq 0.9$, one can observe that BC2 and BC3 leave several unfilled areas along the length of the dogbone. As previously mentioned, the cell under-filling leads to a systematic error in the prediction of quantities of interest. In contrast, BC4 yields a higher local degree of filling, with fewer numerical artifacts due to the incorrect distribution of α, thus leading to results that are closer to a pure polymer state.

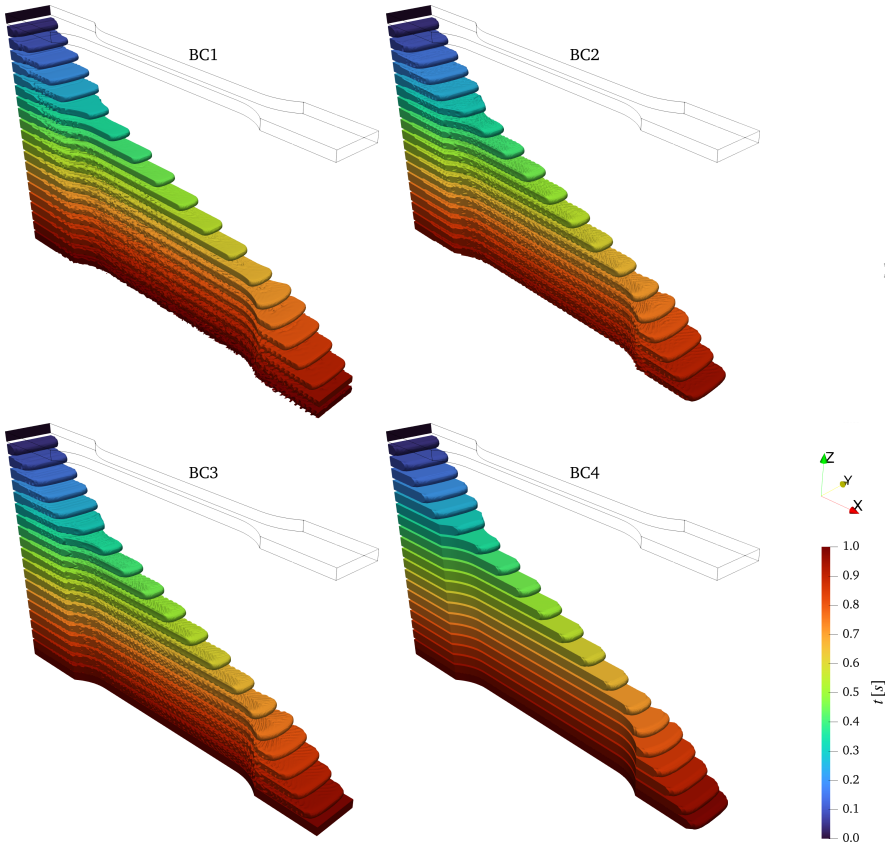


Figure 8.15: Comparison of VoF field for the dogbone specimen as a function of time. Sequence was obtained by clipping the VoF field with $\alpha \geq 0.9$. No-flux boundary conditions were used for BC2-BC4

The pressure history for the conditions using a no-flux boundary condition on α is depicted in Figure 8.16. The figure shows a similar pressure evolution for conditions BC2 and BC3, with the former exhibiting numerical chatter at discrete intervals. The condition BC4 also exhibits numerical chatter but with a smaller amplitude over a longer time interval. However, owing to the higher local degree of filling, this condition shows a steeper pressure gradient than the remaining dynamic outlet conditions. Throughout each simulation, a maximum imbalance of 5×10^{-4} is observed for the BC2, which is visually indistinguishable from the amount of α inserted into the domain when presented in a VoF balance plot.

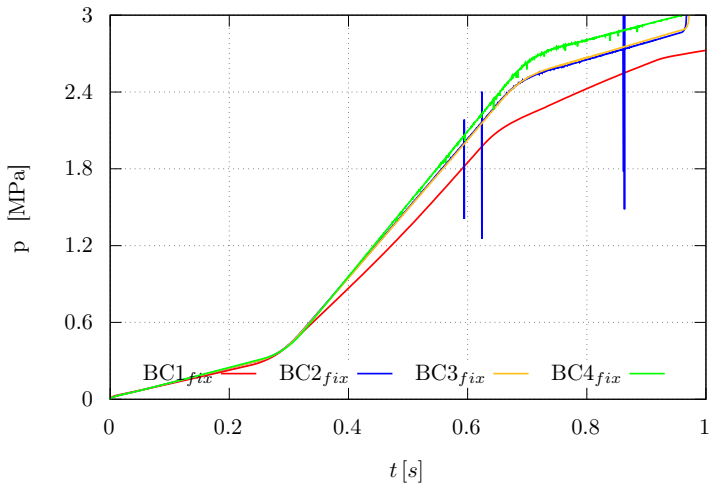


Figure 8.16: Pressure history for BC1-BC3 using the no-flux boundary conditions

From the presented case studies, under incompressible settings, condition BC4 together with the no-flux boundary condition on α, appears to lead to the most realistic filling behavior. However, this numerical setup still requires improvements to avoid numerical chatter on the pressure field. The computational tests were executed in 10 parallel processes through domain decomposition, with a custom implementation of the standard Message Passing Interface (MPI) in foam-extend. The simulations were executed on a workstation with a AMD Ryzen Threadripper 3970X 32-Core processor, with 128 Gb RAM at 2666 MHz, and running Ubuntu 22.04. The calculation times for each condition were as follows: 1041 s for BC1, 1067 s for BC2 (which premature termination), 2097 s for BC3, and 1055 s for BC4.

8.6 Solver validation

To validate the developed solver, its accuracy and performance were compared against the proprietary software Moldex3D® Studio 2021 [2]. This technology also employs the FVM to discretize the governing equations, and the VoF technique for free-surface tracking [84]. Validation tests were performed under both incompressible and compressible flow assumptions.

8.6.1 Spiral

For an incompressible flow case, a spiral geometry was utilized, as depicted in Figure 8.17, where only the inlet condition is specified, with the remaining boundaries treated as a dynamic outlet condition. In this test case, the filling pattern, inlet pressure history, and the velocity profile at discrete locations along the spiral, designated as $p1 - p5$, were compared between both applications.

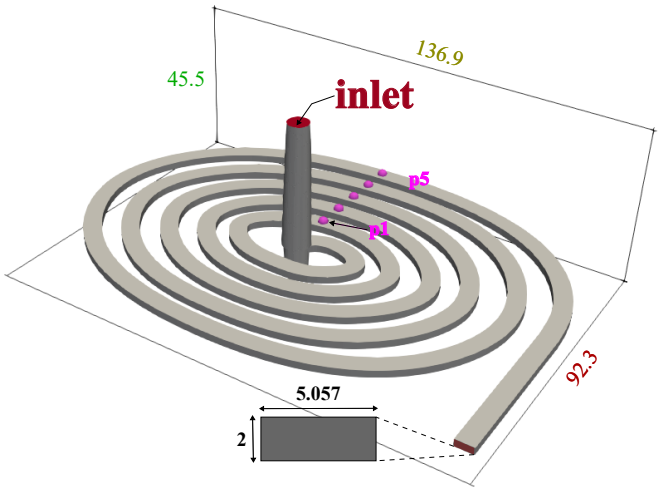


Figure 8.17: Geometry and boundary patches for the spiral case study. Dimensions in millimeters

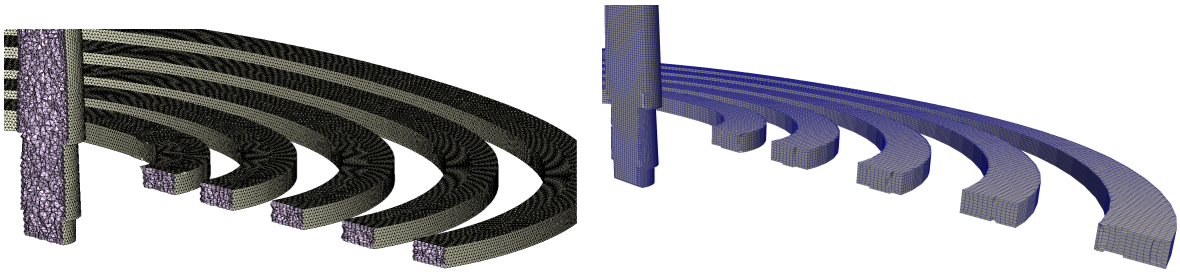
The computational mesh was created with the `cartesianMesh` utility from cfMesh [85]. The final computational domain comprised $411\ 963$ cells, with a maximum orthogonality of $66°$ and maximum skewness of 1.5.

Since Moldex3D® has its own meshing utility, the computational domain was created using its boundary layer mesh (BLM) routine, placing three prismatic layers along the part thickness, connected at the middle region by tetrahedral cells. The final mesh consisted of 3.1 M cells. A detailed view of the clipped computational mesh employed for both software is depicted in Figure 8.18.

To perform incompressible simulations in the proprietary application, a custom material was defined with a constant Newtonian viscosity of 10 Pa · s and a density of 1000 kg/m^3. The fill time was specified to be 2 s with the end of filling at 99% fill.

In Moldex3D®, the user benefits from a streamlined workflow for setting up the IM case, but it encounters limitations when defining custom boundary conditions, linear solver settings, and assessing results from the VoF formulation. At the inlet, a flow rate was prescribed to achieve a filling time of approximately 2 s. The simulation utilized the `enheancedP` solver, excluding `viscous heating`, `non-isothermal`, `non-Newtonian flow` and `compressible flow` from the calculation options and the heat transfer coefficient was set to zero.

In `IMFoam`, the prescribed boundary conditions are rather straightforward. At the inlet patch, a Dirichlet velocity was prescribed to achieve the same fill time as in Moldex3D®, with a null normal



(a) Moldex3D® (b) OpenFOAM®

Figure 8.18: Computational meshes employed for the spiral test case.

gradient boundary condition for pressure. The remaining patches were assigned the linear dynamic outlet, with the no-flux condition for α. The α-lagged solution strategy was employed, and the discretization schemes employed for the governing equations are summarized in Table 8.10. The linear solver settings to solve the discretized system of equations are consistent with those used for the dogbone specimen case study presented in Section 8.5.2. A Courant number of 1 was used to control the time step. The PIMPLE algorithm performed a maximum of 30 outer iterations with 7 PISO corrections. Early exit was allowed when the normalized initial residual was below 5×10^{-4}, 1×10^{-6} for the momentum balance equation and pressure equation, respectively, and the relative change of the inlet pressure between consecutive outer iterations was below 5×10^{-4}. A momentum implicit under-relaxation factor of 0.9 was applied during the outer iterations of the PIMPLE algorithm, reverting to 1 for the final iteration.

The VoF field was initialized with a value of 1 on the layer of cells adjacent to the inlet, representing a global $\alpha = 5 \times 10^4$.

Table 8.10: Discretization schemes for the spiral test case.

Operator / term	Discretization scheme	OpenFOAM terminology
Time derivative, $\partial\phi/\partial t$	Implicit Euler	`Euler`
Gradient, $\nabla\phi$	Cell limited least-squares	`cellMDLimited leastSquares 1`
Convective term :		
$\nabla \cdot (\rho\mathbf{u}\mathbf{u})$	Linear upwind	`Gauss linearUpwindV grad(U)`
$\nabla \cdot (\alpha\mathbf{u})$	Van Leer	`Gauss vanLeer`
$\nabla \cdot (\phi_r\alpha\,(1-\alpha))$	Interface compression	`Gauss interfaceCompression`
$\nabla \cdot \left(\mu\left[(\nabla\mathbf{u})^{\mathrm{T}} - \frac{2}{3}\mathbf{I}\,(\nabla\cdot\mathbf{u})\right]\right)$	Central differencing	`Gauss linear`
Laplacian, $\nabla \cdot (\Gamma\nabla\phi)$	Central differencing with non-orthogonality correction	`Gauss linear corrected`
Interpolation from cell centers to face centers	Linear interpolation	`linear`
Surface-normal gradient, $\nabla_f^{\perp}\phi$	Corrected surface-normal gradient	`corrected`

As observed in Section 8.5.2, during integration process using the no-flux approach, a slight local unboundedness was observed for the VoF field together with pressure field chatter. To mitigate these effects, a distributive algorithm that identifies locally excessive α values and redistributes it uniformly within a core region behind the interface was employed. This approach proved to be less detrimental than the simple clipping of α performed in Section 8.5.2, although at a higher computational cost.

The filling sequence of both applications are depicted in Figures 8.19 and 8.20. Analysis of these images allows to conclude that the filling patterns are in reasonable agreement for both software, although Moldex3D® exhibits a noticeable lack of curvature at the flow front. Overall, the VoF field distribution in `IMFoam` is very satisfactory and remains close to unity throughout most of the part. While small departures from unity can lead to an error in the predicted properties due to the α-weighting blending, the present strategy yields a solution that fairly captures the filling dynamics in IM.

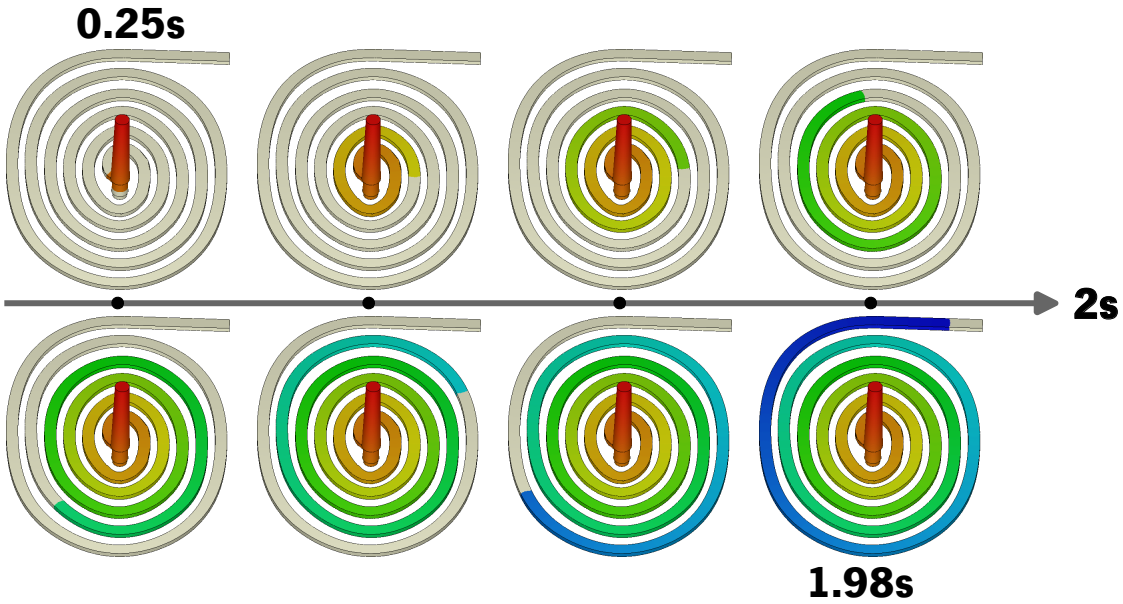


Figure 8.19: Cavity filling sequence for Moldex3D®. Each image represents the fill time at 0.25 s intervals, with the exception of the last one

To infer the quality of the solution, the inlet pressure history and the velocity profile along the thickness of the part at positions $p1 - p5$ are depicted in Figure 8.21 for both applications. The inlet pressure remains relatively low during the filling the sprue region; however, once the melt enters the spiral region, a linear increase in pressure can be observed. Overall, the differences in inlet pressure evolution are within an acceptable range between both applications. The evolution in `IMFoam` was recorded at each time step, showcasing minor amplitude pressure chattering throughout the flow history. This could have be obscured with a smaller sampling-rate. However, for the sake of clarity , the complete pressure signal is reported.

The velocity profiles measured at points $p1 - p5$ show a strong correlation between applications. The results from `IMFoam` overlap for the different sampling locations, whilst some variability can be observed in Moldex3D®. These discrepancies might be caused by potential

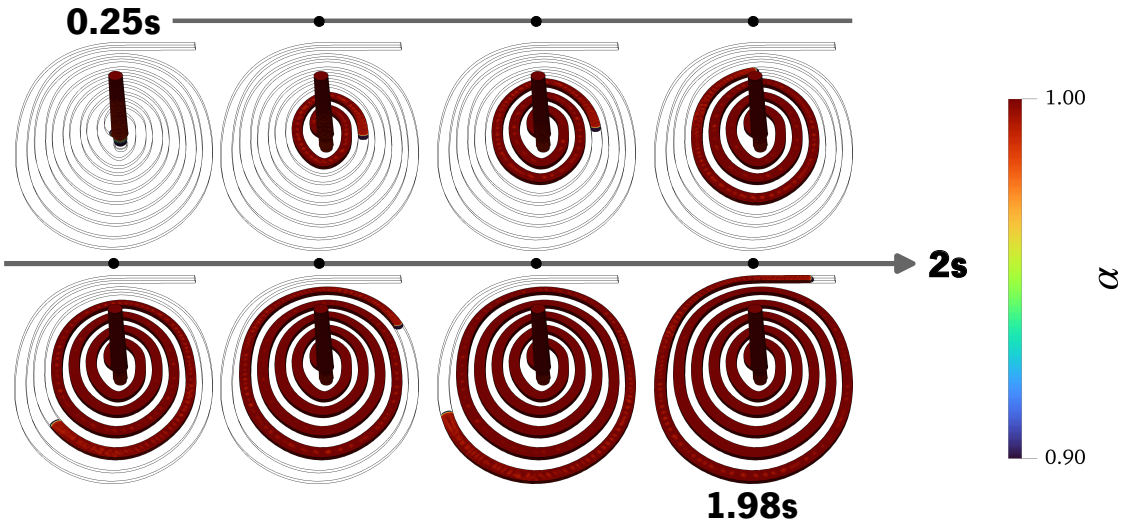
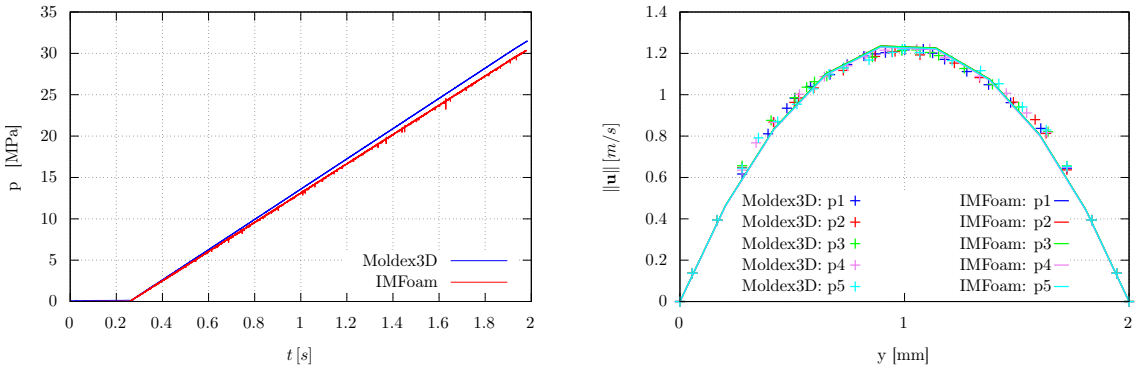


Figure 8.20: Cavity filling sequence for `IMFoam`. Each image represents the fill time at 0.25 s intervals, with the exception of the last. Colormap shows regions where $\alpha \geq 0.9$

differences in mesh conformation at the various sampling locations.



(a) Inlet pressure history

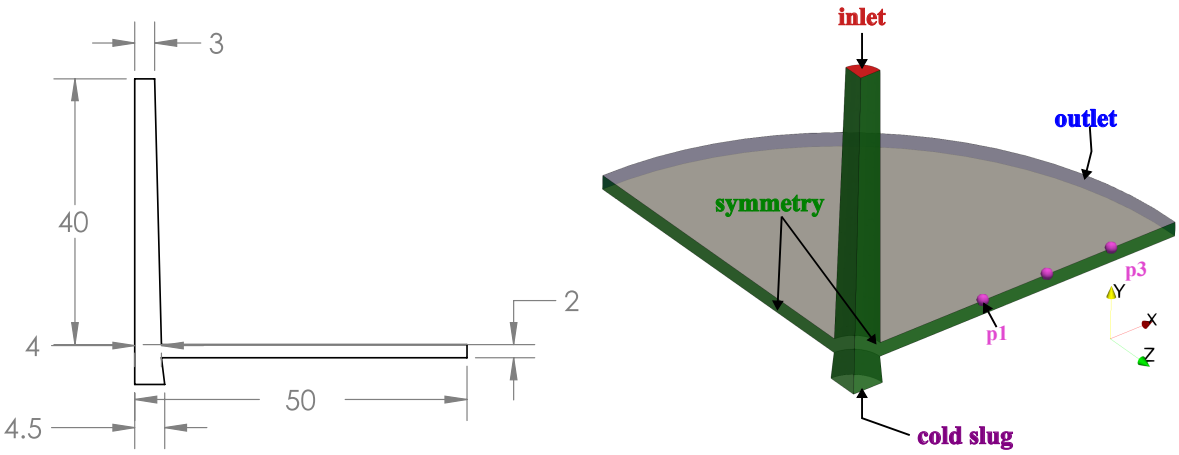
(b) Velocity profile across the thickness at different sampling locations at 99% fill

Figure 8.21: Comparison of results between applications for the spiral test case.

The simulation in OpenFOAM® was executed using 32 parallel processes on a workstation with a AMD Ryzen Threadripper 3970X 32-Core processor, with 128 Gb of RAM (2666 MHz) and running Ubuntu 22.04. The calculation time was $58\,725$ s. The Moldex3D® simulation was executed using 48 threads on a workstation with AMD Ryzen Threadripper 3960X 24-Core processor, with 64 Gb RAM (3200 MHz) and running Windows 11. The calculation time was $50\,379$ s. A direct comparison of the reported computation times is not appropriate, since the simulations were executed on different hardware and software environments, involving distinct processor architectures, core counts, clock speeds, memory resources, operating systems, parallel execution and mesh sizes. Accordingly, the computation times are presented only for completeness purposes.

8.6.2 Center gated disk

For a compressible flow case, a center-gated disk was used. The geometry and corresponding boundaries are illustrated in Figure 8.22. Due to the symmetry of the geometry, only one-fourth of the part was considered in the calculation. For this test case, the filling pattern, inlet pressure history, velocity at positions $p1 - p3$ (at $x = 20$, $x = 30$, and $x = 40$ mm), and the pressure variation from $x = 20$ to $x = 50$ mm were compared between the two applications.



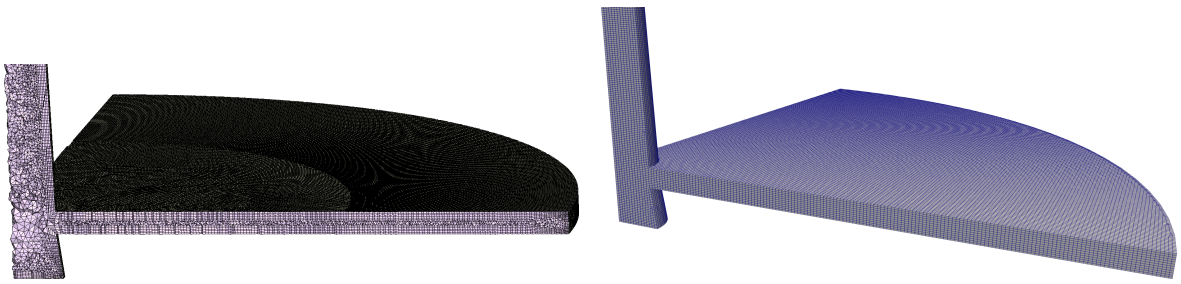
(a) CGD cross-section geometry (dimensions in millimeters)

(b) CGD patch names

Figure 8.22: Geometry and boundary patches for the CGD case study

The computational domain of the CGD was created with the `cartesianMesh` [85]. The final computational domain comprised $288 \;\; 128$ cells, with 12 along the cross-sectional thickness, a maximum non-orthogonality of $52°$and a maximum skewness of 1.1.

Using Moldex3D® BLM technology, the computational domain was generated by placing 5 prismatic layers along the thickness of the part, connected at the middle region by tetrahedral cells. The final mesh consisted of 7.3 M elements, with 15 elements across the part thickness. A detailed view of the mesh used in each software is depicted in Figure 8.23.



(a) Clipped Moldex3D® mesh

(b) OpenFOAM®

Figure 8.23: Computational domain for the CGD test case.

On the commercial application, at the inlet, a flow rate was prescribed to achieve a filling time

of approximately 1 s. For the temperature, a Dirichlet condition of 513.15 K was defined for the inlet, and for the remaining boundaries a convective heat flux (CHT) was prescribed, considering an ambient temperature of 323.15 K and a heat transfer coefficient of $1\,250$ w/(m$^2\cdot$ K). The governing equations were solved with the `Enhanced-P` solver enabled, and by considering: `viscous heating`, `non-isothermal`, `stabilized calculation`, `non-Newtonian flow` and `compressible flow`, together with the `accurate` setting enabled and a stopping criteria defined at the default 99.95%.

The prescribed boundary conditions for the OpenFOAM® setup are detailed in Table 8.12. At the inlet, a time-varying Dirichlet boundary condition was applied to the velocity, utilizing the flow rate profile obtained from Moldex3D®. A no-slip boundary condition was assigned at the walls, with a null normal gradient prescribed at the outlet, and a dynamic outlet with a sigmoidal transition to the cold slug. For the temperature field, a fixed temperature was prescribed at the inlet, a Robin-type boundary condition was applied at the walls by specifying the ambient temperature and the heat transfer coefficient, using the same values as in its proprietary counterpart. For the cold-slug and outlet, a null normal gradient condition was assigned.

A literature survey [84, 86] indicates that Moldex3D® only accounts for shear-heating effects in the temperature equation, thus, for this test, pressure-work heating was neglected from Equation (8.3) (the first term on the right-hand side). The numerical integration in OpenFOAM® was performed from $[0-1]$ s, using a Courant number of 0.5. The discretization schemes are summarized in Table 8.11, and the solution algorithms were identical to the ones in Table 8.8 but using a relative tolerance of 0.1 for both velocity and temperature.

Table 8.11: Discretization schemes for the CGD.

Operator / term	Discretization scheme	OpenFOAM terminology
Time derivative, $\partial\phi/\partial t$	implicit Euler	`Euler`
Gradient, $\nabla\phi$	cell limited least-squares	`cellLimited leastSquares 1`
Convective term :		
$\nabla\cdot(\rho\mathbf{uu})$	Limited linear	`Gauss limitedLinearV 1`
$\nabla\cdot(\rho\mathbf{u}T)$	Limited linear	`Gauss limitedLinear 1`
$\nabla\cdot(\alpha\mathbf{u})$	Van Leer	`Gauss vanLeer`
$\nabla\cdot(\phi_r\alpha\,(1-\alpha))$	interface compression	`Gauss interfaceCompression`
$\nabla\cdot\left(\mu\left[(\nabla\mathbf{u})^{\mathrm{T}}-\frac{2}{3}\mathbf{I}\,(\nabla\cdot\mathbf{u})\right]\right)$	Central differencing	`Gauss linear`
Laplacian, $\nabla\cdot(\Gamma\nabla\phi)$	Central differencing with non-orthogonality correction	`Gauss linear limited 1`
Interpolation from cell centers to face centers	Linear interpolation	`linear`
Surface-normal gradient, $\nabla_f^{\perp}\phi$	Corrected surface-normal gradient	`limited 1`

Table 8.12: Boundary conditions used in three dimensional CGD case. The acronym zG stands for `zeroGradient`, fV for `fixedValue`, dO for `dynamicOulet`, CHT for `convectiveHeatFlux`

	$\mathbf{u}$	p	T	α
inlet	*fV*	*zG*	*fV*	*fV*
outlet	*zG*	*fV*	*zG*	*zG*
walls	*fV*	*zG*	*CHT*	*zG*
coldSlug	*dO* [sigmoidal]	*dO*[sigmoidal]	*zG*	*zG*

The PIMPLE algorithm, using the standard solution approach, was used to enforce pressure-velocity coupling. The solution strategy performed 20 outer iterations with 4 PISO correction loops per outer iteration. Early exit was allowed if the initial normalized residuals for the velocity, pressure, and temperature fell bellow 1×10^{-4}, 1×10^{-5} and 1×10^{-5}, respectively. Similarly, to the previous cases an implicit under-relaxation factor 0.9 was applied to the momentum balance during the outer iterations. For the energy conservation equation, an implicit under-relaxation of 0.99 was used.

A general purpose Polypropylene, Moplen HP 500N, was chosen for the IM filling simulation. Its rheological and equation-of-state model parameters are reported in Table 8.13, and were used in both software.

Table 8.13: Rheological and Equation-of-state parameters for PP Moplen HP 500N

Rheological Model				Equation-of-state			
Cross-WLF	n	0.344589	[-]	Modified Tait	b1L	0.00124681	$[\mathrm{m}^3/\mathrm{Kg}]$
	τ^*	18665.5	[Pa]		b2L	1.09183×10^{-6}	$[\mathrm{m}^3/(\mathrm{Kg} \cdot \mathrm{K})]$
	D1	6.69×10^{11}	[Pa · s]		b3L	1.06×10^{8}	[Pa]
	D2	263.15	[K]		b4L	0.00623459	[1/K]
	D3	0.0	[K/Pa]		b1S	0.0011394	$[\mathrm{m}^3/\mathrm{Kg}]$
	A1	25.2305	[-]		b2S	4.28268×10^{-7}	$[\mathrm{m}^3/(\mathrm{Kg} \cdot \mathrm{K})]$
	$\tilde{A}_2$	51.6	[K]		b3S	2.13×10^{8}	[Pa]
					b4S	0.00453208	[1/K]
					b5	393.15	[K]
					b6	1×10^{-7}	[K/Pa]
					b7	5.91681×10^{-5}	$[\mathrm{m}^3/\mathrm{Kg}]$
					b8	0.1	[1/K]
					b9	1×10^{-8}	[1/Pa]

The developed solver currently assumes constant values for the heat capacity and thermal conductivity. Consequently, the material properties of the chosen thermoplastic were modified in Moldex3D® to have constant values of $C_p = 2400$ J/(kg · K) and $k = 0.15$ W/(m · K).

The filling sequence of both simulations is depicted in Figure 8.24 and Figure 8.25. Analysis of Figure 8.24 shows a lack of interface curvature in the sprue region. During the filling of the disk region, the profile remains comparatively flat, with the part not being completely filled at 1 s. Given that the interface clipping criterion and boundary conditions of the proprietary software are not provided nor adjusted by the user, the results are interpreted qualitatively. Figure 8.25 shows a smoother polymer profile. At the end of filling, the distribution of α in the domain seems to be different from the proprietary software, with a wide area showing $0.9 < \alpha \leq 1$. As seen in Section 8.5.1, these local under-filled regions contribute to deviations from the expected polymer properties, suggesting that the use of no-slip conditions on the pseudo-air, although not as problematic as under incompressible settings, may prevent the cavity from becoming fully filled.

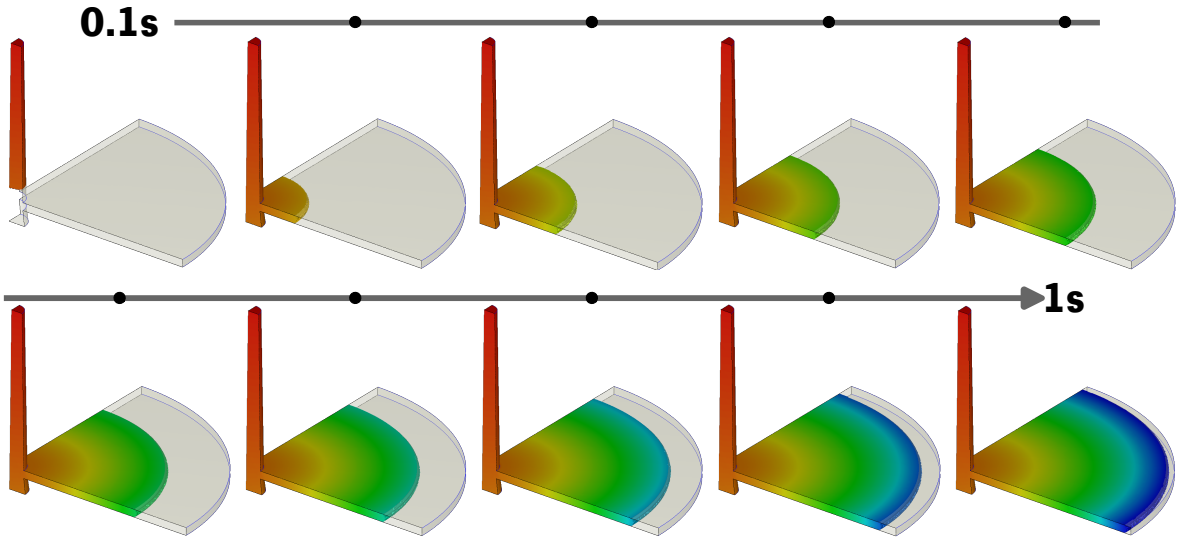


Figure 8.24: Fill sequence of Moldex3D®. Each image is represent the fill time at 0.1 s intervals.

To assess the differences in the predicted fields between both applications, we compare the pressure drop at the end of filling (1 s), both globally and along the radial direction from $20 - 50$ mm, as well as the velocity profile along the thickness of the disk at $x = 20$, $x = 30$, and $x = 40$ mm, together with the pressure history at the inlet. Given the symmetry in the CGD, only one-quarter of the part is shown in Moldex3D®.

A comparison between the global pressure drop obtained in both applications is depicted in Figure 8.26 at 1 s.

Under the chosen settings, the developed application predicts a slightly higher pressure level than the proprietary software. Given that Moldex3D® appears to predict a pressure below atmospheric pressure, we assume that this quantity needs to be added to the computed values. Thus, a maximum difference < 0.2 MPa is observed in the sprue region between both applications.

Figure 8.27 depicts the velocity and the pressure profiles at the sampling locations. A comparison of the obtained velocity profiles reveals that the greatest difference occurs at $x = 20$ mm, with the numerical difference decreasing along the radial direction. The pressure profile obtained with `IMFoam`

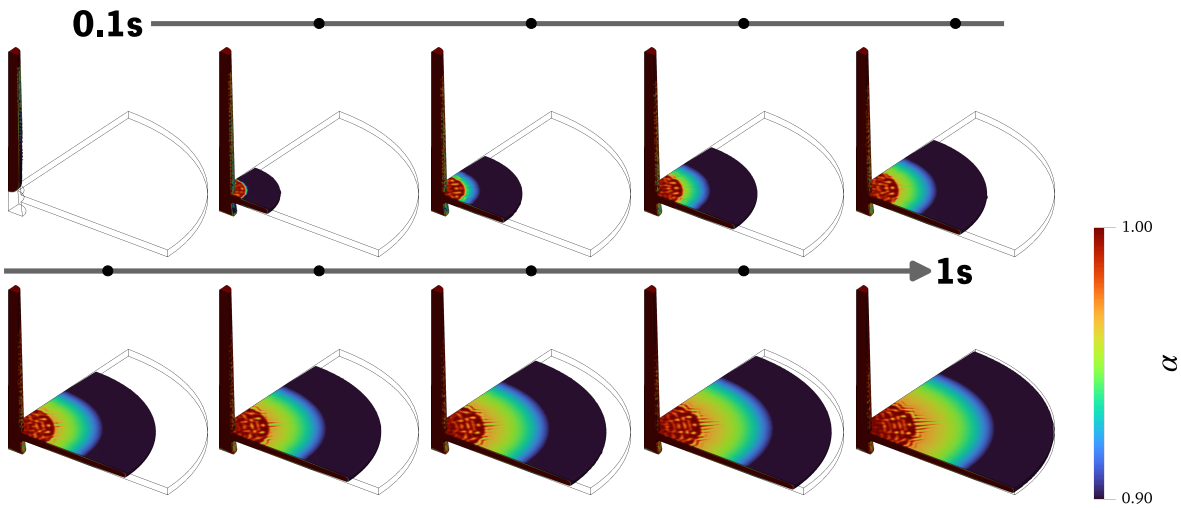
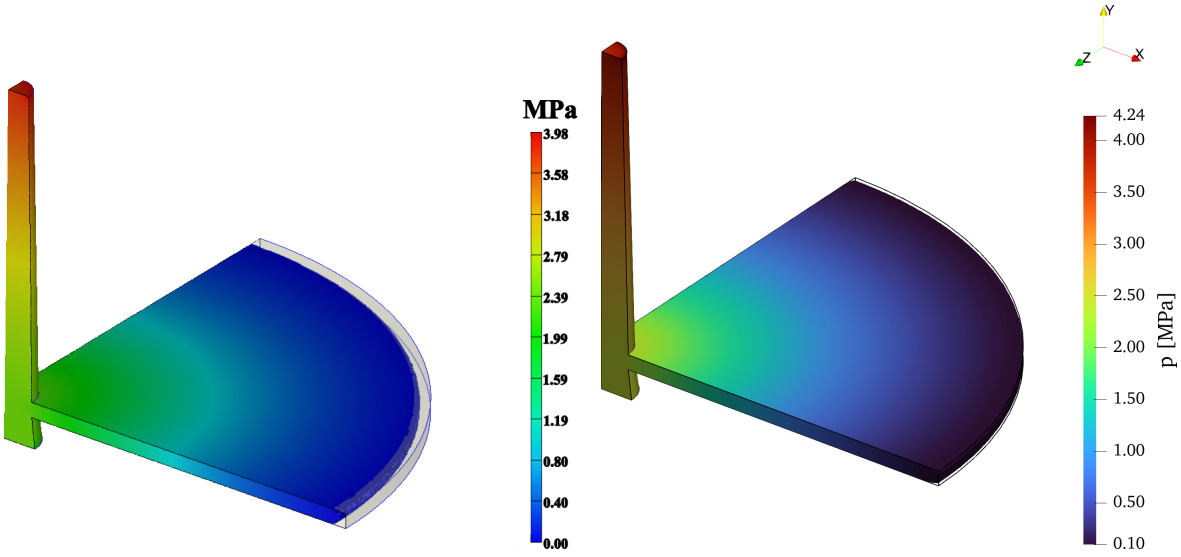


Figure 8.25: Fill sequence of `IMFoam`. Each image is represent the fill time at 0.1 s intervals. Colormap shows regions where $\alpha \geq 0.9$



(a) Moldex3D®

(b) `IMFoam`

Figure 8.26: Pressure field at the $t = 1$ s

matches reasonably well with the predictions of the proprietary software.

The inlet pressure history is depicted in Figure 8.28. Analysis of the figure shows an irregular pressure evolution at the beginning of the simulation, coincident with the transition from the sprue to the disk section in the commercial application. Moreover, as observed in Pedro et al. [8] Moldex3D® predicts oscillations that cannot be correlated to any of the phases of the filling stage. The inlet pressure evolution is smoother for `IMFoam`, however, minor pressure oscillations at the inlet can be observed, which are attributed to the dynamic outlet condition used, as discussed in detail in the previous section.

Given the transient, non-Newtonian, and non-isothermal nature of the simulation, we consider the observed differences to be within an acceptable range. The main observed difference is the required computational time. The proprietary software was executed in parallel with 48 threads using

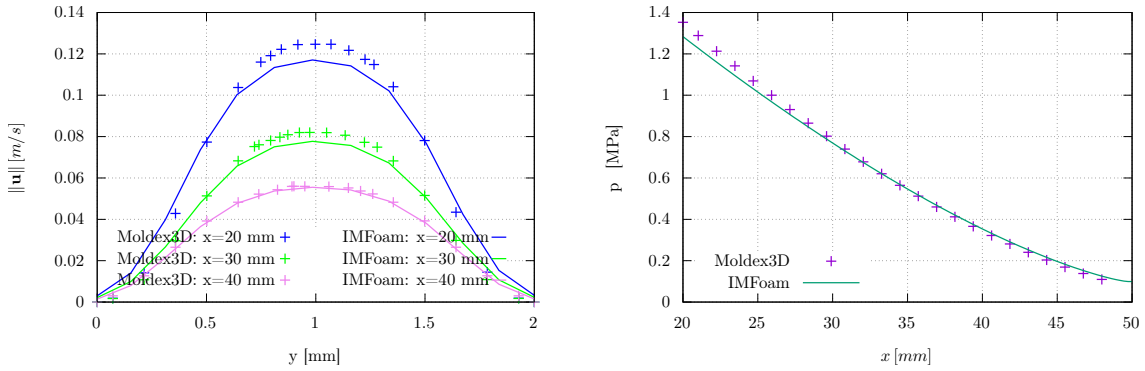


Figure 8.27: Comparison of the computed velocity profile at three distinct locations along the radius of the CGD (left), and pressure along the radius (right) at $t = 1$ s. Symbols are data collected from Moldex3D®, lines are from OpenFOAM®

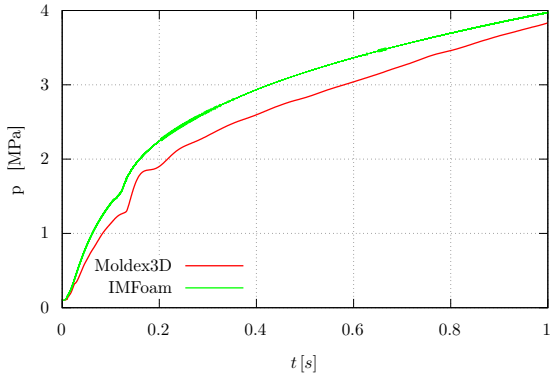


Figure 8.28: Comparison of the the pressure history between Moldex3D® and OpenFOAM®

a workstation with a AMD Ryzen Threadripper 3960X 24-Core processor, with 64 Gb RAM at 3200 MHz, running Windows 11, on the full geometry and took $\approx 7.5\ h$ to reach 1 s. The OpenFOAM® solver, using one-quarter of the geometry, with the chosen settings, took $\approx 29\ h$ using a workstation with a AMD Ryzen Threadripper 3970X 32-Core processor, with 128 Gb RAM at 2666 MHz, running Ubuntu 22.04. A direct comparison of these computation times again not appropriate, due to the different settings. Accordingly, the reported computation times are provided solely for completeness. Anyway, the solution strategy used by Moldex3D® is not provided to the user, making it unclear what boundary conditions are being used, how the governing equations are actually being solved, which Courant number is used, and what tolerance values are being considered.

8.6.3 Fiber orientation modeling

Following the utility developed in Chapter 7, the filling and orientation modeling have been assessed in a fan-gated disk (FGD) . Figure 8.29 depicts the overall dimensions of the part, the corresponding patches, and measurement locations for the velocity and orientation profiles. For this case, only the inlet is specified, with the remaining patches treated as a dynamic outlet, in

accordance with the approach described in Section 8.6.1. The inlet pressure history, velocity profile, and orientation profile at locations $p1 - p3$, at ($x = 0$ mm, $z = 19.5$ mm, $\bar{y}_{end} = \{51, 31, 11\}$ mm, where $\bar{y}_{end}$ denotes the y-coordinate measured relative to the end of the disk, are compared between OpenFOAM® and Moldex3D®.

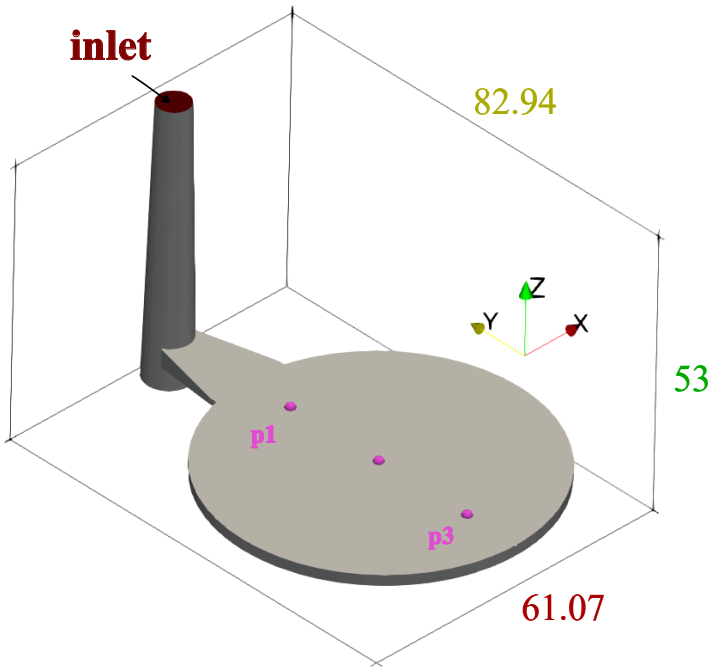
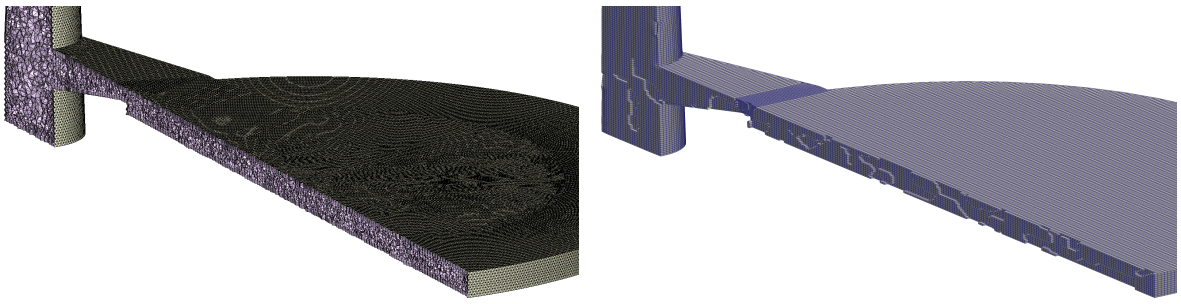


Figure 8.29: Geometry and boundary patches of the FGD study. Dimensions in millimeters

The computational domain in OpenFOAM® was generated with the `cartesianMesh` [85] utility, resulting in a mesh with $477\,259$ cells, with a maximum non-orthogonality of $64.4°$ and a maximum skewness of 2.0. In Moldex3D®, the BLM technology was used to place 3 boundary layers along the surface of the part, resulting in computational domain with 4.3 M cells. A comparison between the two computational domains is depicted in Figure 8.30.



(a) Moldex3D®

(b) OpenFOAM®

Figure 8.30: Computational domain for the fan gate case

Assuming fluid incompressibility, the filling conditions were set to fill the part in 1 s and to stop at 99% fill. The viscosity of the polymer phase was set to 100 Pa · s, and the density to 1000 kg/m^3 with the pseudo-air properties scaled by a factor of 1×10^{-3} to the ones of the polymer counterpart.

In OpenFOAM®, the same settings as in Section 8.6.1 were applied, with the additional discretization scheme for $\nabla \cdot (\mathbf{u}\mathbf{A}_2)$ set to first-order upwind scheme, and the orientation dynamics equation using the isotropic rotary diffusion model, Listing 7.1, with the Invariant Based Optimal Fitting [87] closure, solved through with BiCGStab as a linear solver with DILU as a preconditioner with an absolute tolerance of 1×10^{-12}. A value of unity was used for the fiber form factor and 0.01 for the phenomenological fiber-fiber interaction parameter. The orientation equation was allowed to evolve only in cells with $\alpha \geq 0.5$ thorough a simple binary mask applied to the source term. The orientation dynamics were advanced after solving the pressure-velocity system and advection of the interface, within the α-lagged strategy. The decoupled momentum-orientation system was corrected five times per time step, being normalized by the trace of the second-order orientation tensor in each correction loop. The condition for early exit of the pressure equation was also modified to terminate when the initial residual fell below 1×10^{-5}.

In Moldex3D®, the same orientation model was selected. However, to set the form factor to a value near unity, the fiber aspect ratio was set to 100. Deviating from a unitary value produces a periodic Jeffery orbit when shear effects dominate, with fibers predominantly aligned in the flow direction [88].

The predicted filling sequence for both applications is depicted in Figure 8.31 for Moldex3D® and in Figure 8.32 for `IMFoam` using the criterion $\alpha \geq 0.9$. Analysis of both images reveals reasonable agreement between the filling progression predicted by both software. Consistent with previous simulations, Moldex3D® exhibits a lack of curvature at the flow front, while this feature is better captured in `IMFoam`. Examination of the VoF field in Figure 8.32 indicates that the phase-fraction distribution during filling is not strictly unitary across the filled region. This results in a weighted property distribution that does not correspond to a pure polymer state. However, the obtained VoF field is considered to provide a reasonable approximation. Improving this aspect should lead to a more accurate representation of the filling dynamics using the VoF approach.

The comparison between the predicted inlet pressure history and velocity profiles at positions $p1 - p3$ is depicted in Figure 8.33. The evolution of the inlet pressure predicted in `IMFoam` is in close agreement with that of the commercial counterpart up to $t = 0.3$ s, corresponding to the filling of the sprue and runner. Near this point, Moldex3D® shows a non-physical numerical artifact in the pressure evolution. As filling progresses, the predicted pressure is slightly higher in Moldex3D® than in `IMFoam`, with both exhibiting similar features.

Both application predict a parabolic velocity profile, with the highest values are obtained at position $p1$, followed by $p3$, and finally $p2$. A close match is obtained between both applications, with a greater difference noted at position $p2$. Up to position $p2$ the flow is radially diverging, thus it decreases with distance. At position $p3$, the flow ceases to diverge due the influence of the disk boundary, leading to a converging flow dynamics and an increase in velocity.

The comparison of the orientation profiles at positions $p1 - p3$ is depicted in Figure 8.34.

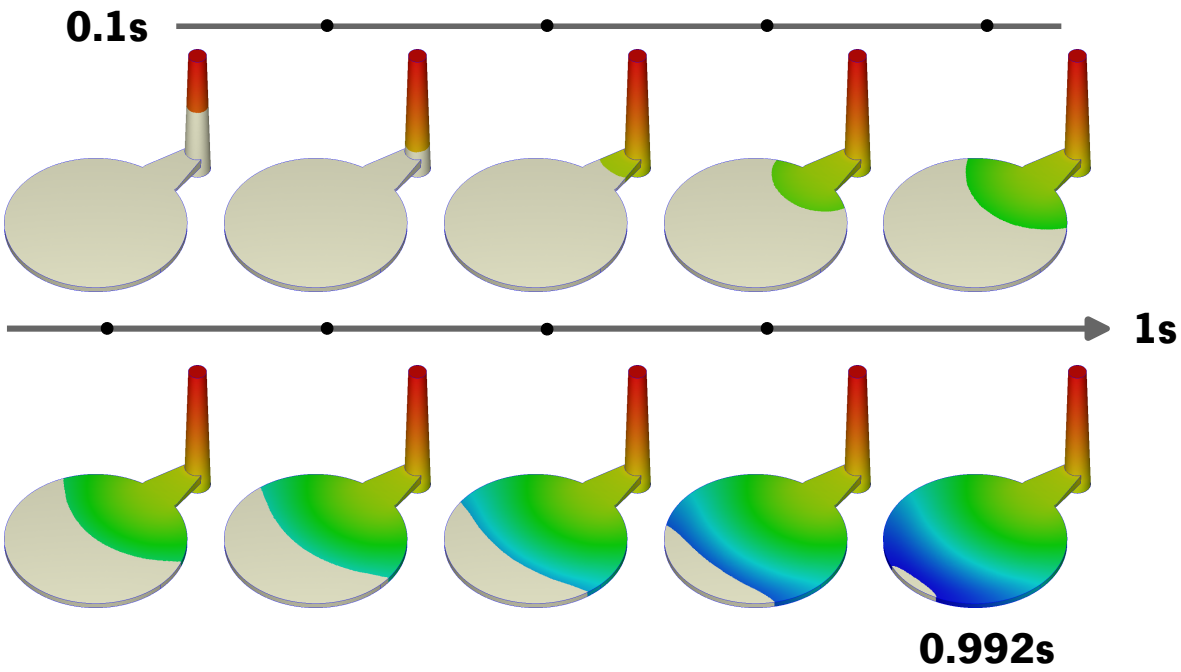


Figure 8.31: Fill sequence of Moldex3D®. Each image is represent the fill time at 0.1 s intervals.

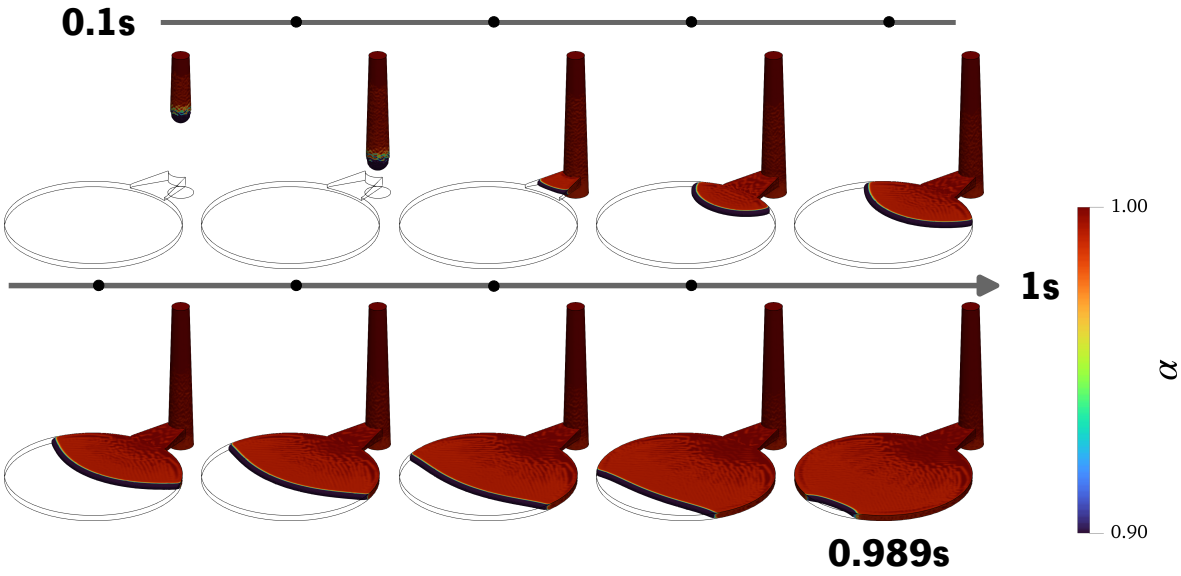
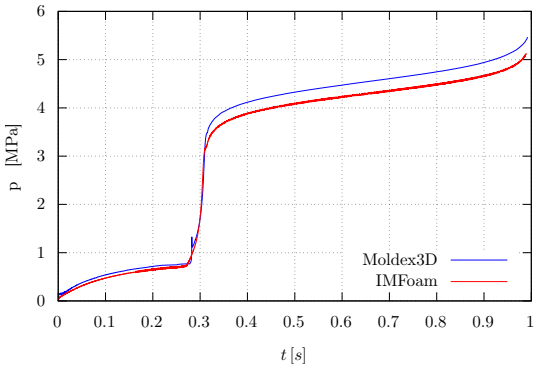
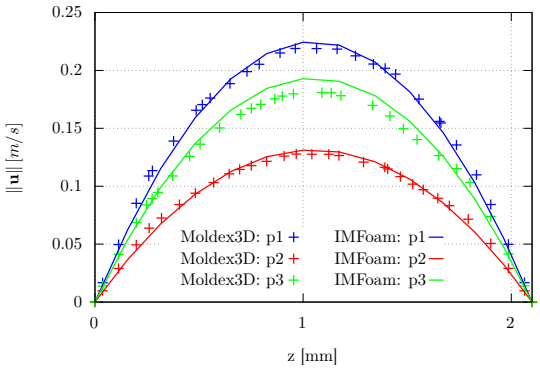


Figure 8.32: Fill sequence of `IMFoam`. Each image is represent the fill time at 0.1 s intervals.

The results obtained with `IMFoam` at positions $p1$ and $p2$ show that the A_{22} component (along the flow direction) is consistent with known literature for a radially diverging flow [88]. At these sampling locations, the degree of orientation is higher near the walls and decreases toward the middle of the part, where flow-direction alignment is less pronounced. In this core region, cross-flow alignment is evident, as shown by the increase of the A_{11} component, which also aligns with known literature [88]. For the orientation profile measured at $p3$, a similar orientation behavior is observed, despite the existence of some converging flow. A similar behavior is observed for the prediction of Moldex3D® at locations $p1$ and $p2$, albeit with less pronounced features. At location $p3$, the orientation predicted by Moldex3D® is rather flat, indicating that it is mainly aligned along the flow direction, with minimal loss of alignment



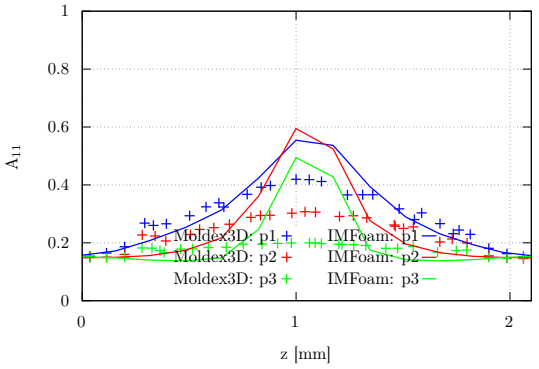
(a) Inlet pressure history



(b) Velocity profile across the thickness at the different sampling locations at 99% fill

Figure 8.33: Comparison of results between applications for the FGD test case

toward the center of the part.



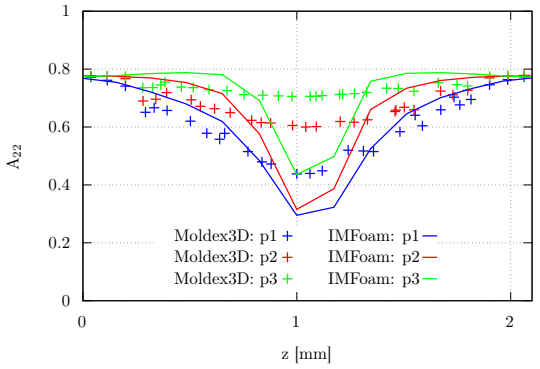


Figure 8.34: Comparison of orientation results between applications for the FGD test case at 99% fill.

The simulation in OpenFOAM® was executed using 32 parallel processes on a workstation equipped with a AMD Ryzen Threadripper 3970X 32-Core processor, with 128 Gb of RAM (2666 MHz) and running Ubuntu 22.04. The total calculation time was $73\,012$ s. The Moldex3D® simulation was performed using 48 threads on a workstation with AMD Ryzen Threadripper 3960X 24-Core processor, with 64 GB of RAM (3200 MHz), and running Windows 11. The calculation time was $16\,187$ s.

8.7 Conclusion

A numerical tool, named `IMFoam`, has been developed within the community-driven fork of OpenFOAM®, aimed at democratizing open-source resources for modeling the injection molding process.

The tool accommodates both incompressible and compressible formulations, offering two algebraic Volume-of-Fluid (VoF) variants and one geometric VoF method for interface capturing. It also

includes dynamic outlet boundary conditions, alongside appropriate rheological models and equations of state.

Verification of the tool was conducted using the Method of Manufactured Solutions to ensure the correct implementation of the governing equations and the associated rheological and equation-of-state models.

Pure advection tests were executed to evaluate the performance of the various VoF approaches. In a vortical flow scenario, the geometric variant demonstrated superior shape retention when compared to the algebraic counterparts, albeit at a higher computational cost. The implicit algebraic variant achieved the lowest computational expenditure, while the Multi-dimensional Universal Limiter for Explicit Solutions (MULES) performed intermediate to these two methodologies. The selection of an algorithm must consider the desired sharpness of the interface and the boundedness of the transported VoF variable. Although faster, the implicit variant was found to be more susceptible to generating unbounded solutions than MULES in three-dimensional examples.

Boundary conditions that selectively allow the secondary phase to exit the domain were proposed and assessed. Numerical tests, conducted in both two and three dimensional case studies, involved analyzing the filling patterns of a planar flow in a center-gated disk cross-section and a tensile dogbone specimen, respectively. In both cases, the assumption of fluid incompressibility was made, along with an inelastic rheological behavior with constant Newtonian viscosity. The developed boundary conditions yielded filling patterns that closely align with known experimental observations. However, the outlet characteristics of the boundaries resulted in mass loss during the filling stage, as well as the potential for back flow. Depending on the selected boundary conditions, mass loss can be significant. To mitigate this issue, a no-flux condition was imposed on the phase indicator governing equation, preventing the primary phase from exiting the domain. This adjustment was pivotal in enhancing filling patterns while addressing the mass loss challenge. However, numerical pressure chatter was observed under certain conditions, indicating that further refinement is necessary to accurately capture dynamic outlet behavior while maintaining a stable pressure evolution. This chatter can be reduced if the calculation tolerances are tighter, thus at a cost of additional computational time.

Lastly, a validation cases were performed by comparing the filling evolution, velocity profiles at discrete locations and inlet pressure evolution within for three-dimensional case studies. The results obtained with the developed application, `IMFoam`, and Moldex3D®, a proprietary software widely used in injection molding industry. The findings show the predictions of `IMFoam` and the proprietary application are in reasonable agreement. However, notable differences in computational time were observed, with Moldex3D® performing significantly faster than the developed solver. In what concerns fiber orientation evolution there are still some discrepancies between both applications, albeit displaying a similar trend. Thus, reducing computational time represents a key direction for future work to enhance the overall efficiency and usability of the developed computational tool. Moreover, the evolution of fiber orientation in multiphase flows should be also studied in more detail.

Appendices

Appendix A Standard solution approach

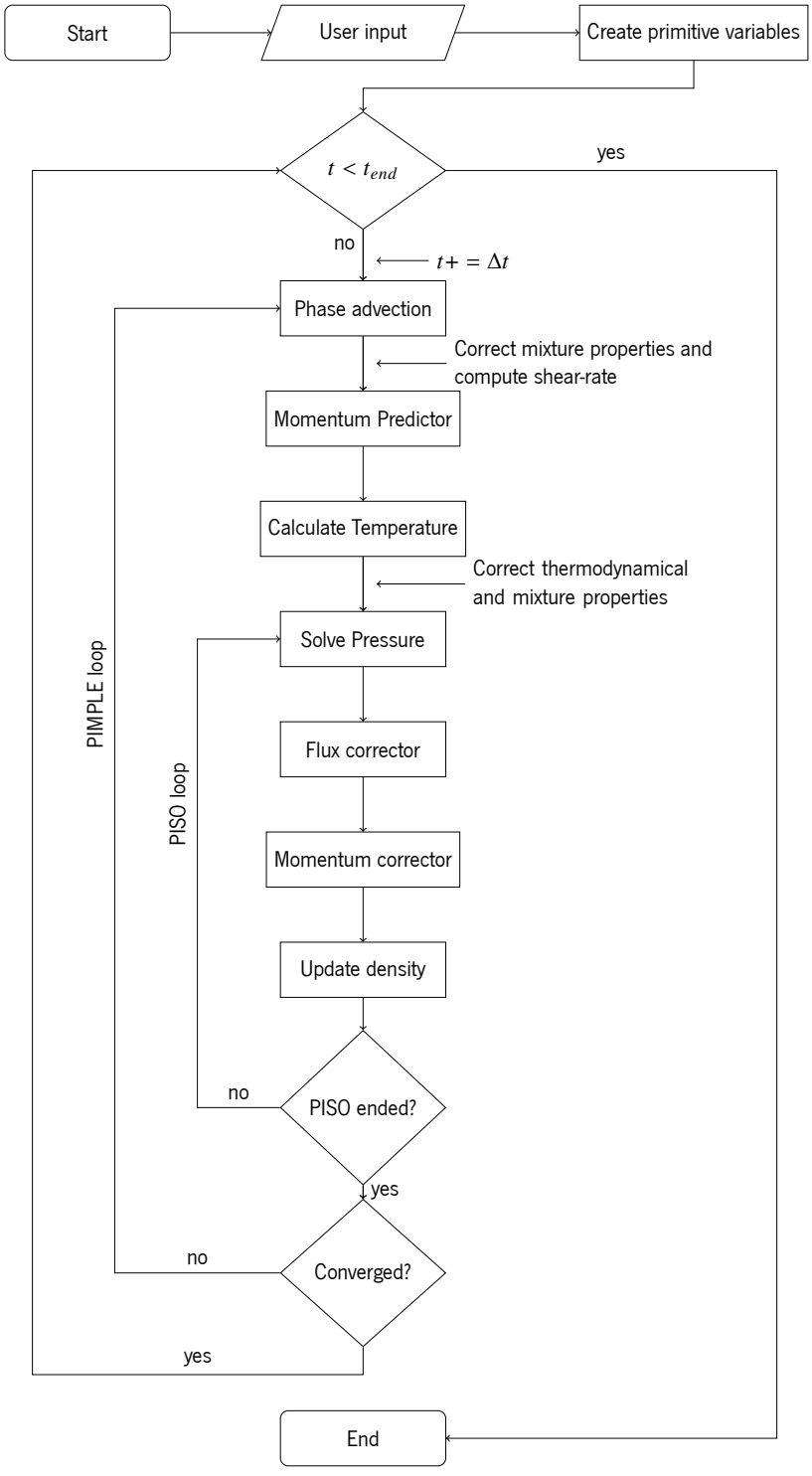


Figure 35: Flow chart of solution algorithm with standard approach

Appendix B α-lagged solution approach

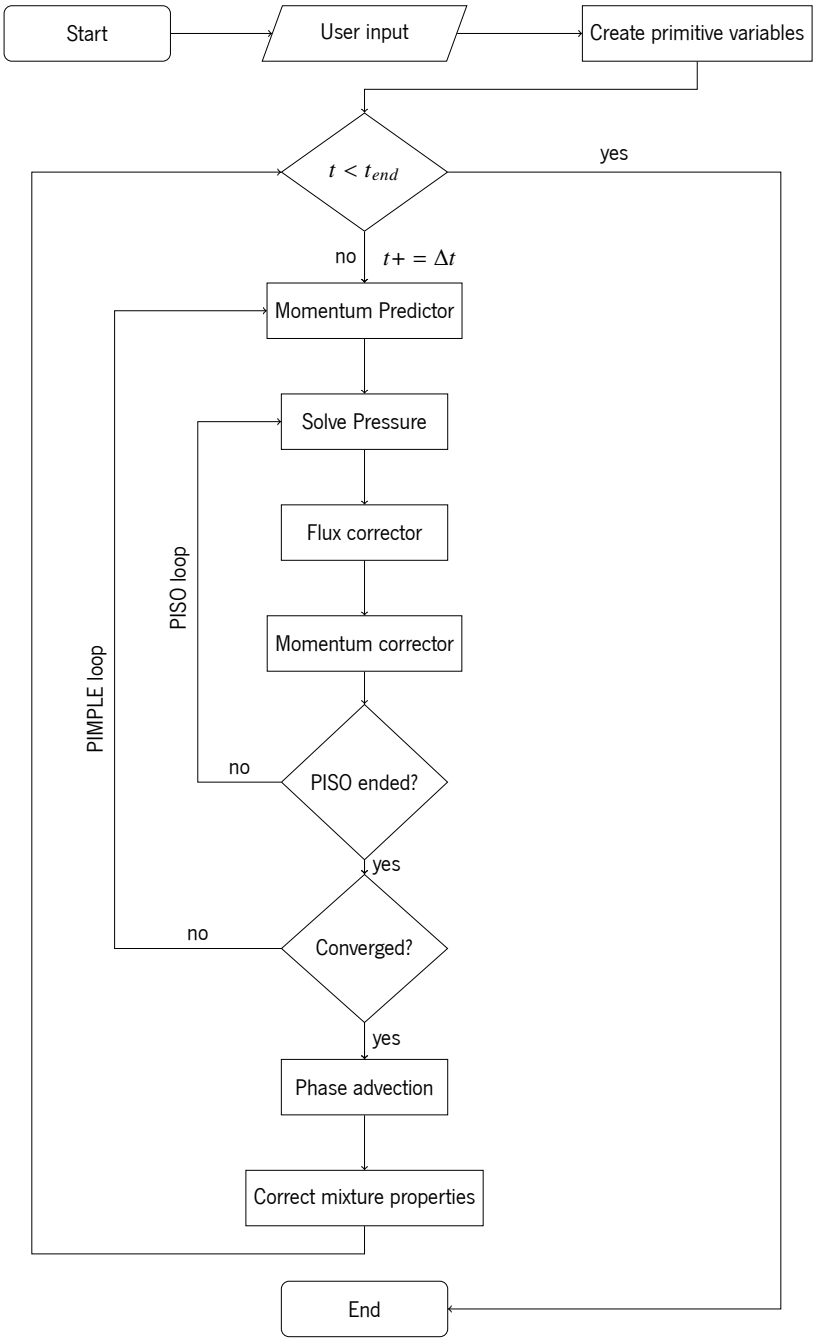


Figure 36: Flow chart of solution algorithm with α-lagged approach

Appendix C Availability of data and material

The data that support the findings of this study are openly available at `https://github.com/Computational-Rheology/IMFOAM`.

References

[1] P.K. Kennedy and R. Zheng. *Flow Analysis of Injection Molds*. Hanser Publishers, 2013.

[2] Moldex3D. `https://www.moldex3d.com/`. Accessed: 2024-10-28.

[3] MoldFlow. `https://help.autodesk.com/view/MFIA/2024/ENU/`. Accessed: 2024-10-28.

[4] CADMOULD. `https://www.simcon.com/cadmould`. Accessed: 2024-10-28.

[5] Inspire Mold. `https://altair.com/inspire-mold`. Accessed: 2024-10-28.

[6] SOLIDWORKS Plastics. `https://www.solidworks.com/media/introduction-injection-molding-simulation`. Accessed: 2024-10-28.

[7] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computer in Physics*, 12(6):620–631, 11 1998.

[8] João Pedro, Bruno Ramôa, João Miguel Nóbrega, and Célio Fernandes. Verification and Validation of openInjMoldSim, an Open-Source Solver to Model the Filling Stage of Thermoplastic Injection Molding. *Fluids*, 5(2), 2020.

[9] João Vidal and João Miguel Nóbrega. An Enhanced Temperature Control Approach to Simulate Profile Extrusion. *Polymers*, 16(7), 2024.

[10] João Castro, João Miguel Nóbrega, and Ricardo Costa. Computational Framework to Model the Selective Laser Sintering Process. *Materials*, 17(8), 2024.

[11] Wagner de Campos Galuppo, Pedro Santana, Francisco Alves, and João Miguel Nóbrega. Extending the Finite Area Method for enhanced simulation of deformable membranes and its application to extrusion blow moulding. *Thin-Walled Structures*, 203:112184, 2024.

[12] Felix Ospald. Numerical Simulation of Injection Molding using OpenFOAM. *PAMM*, 14(1):673–674, 2014.

[13] Jitao Liu, Shuxiang Zhang, Guoqun Zhao, and Guilong Wang. Numerical and experimental investigations on polymer melt flow phenomenon in a vario-thermal mold cavity. *Journal of Applied Polymer Science*, 134(33):45193, 2017.

[14] Wagner de Campos Galuppo, Ana Magalhães, Luís Lima Ferrás, João Miguel Nóbrega, and Célio Fernandes. New boundary conditions for simulating the filling stage of the injection molding process. *Eng. Comput. (Swansea)*, 38(2):762–778, July 2020.

[15] Florian Wittemann, Robert Maertens, Alexander Bernath, Martin Hohberg, Luise Kärger, and Frank Henning. Simulation of Reinforced Reactive Injection Molding with the Finite Volume Method. *Journal of Composites Science*, 2(1), 2018.

[16] Johan Roenby, Henrik Bredmose, and Hrvoje Jasak. A computational method for sharp interface advection. *Royal Society Open Science*, 3(11):160405, 2016.

[17] Henning Scheufler and Johan Roenby. Accurate and efficient surface reconstruction from volume fraction data on general meshes. *Journal of Computational Physics*, 383:1–23, 2019.

[18] Lionel Gamet, Marco Scala, Johan Roenby, Henning Scheufler, and Jean-Lou Pierson. Validation of volume-of-fluid OpenFOAM® isoAdvector solvers using single bubble benchmarks. *Computers & Fluids*, 213:104722, 2020.

[19] Tobias Karl, Jan Zartmann, Simon Dalpke, Davide Gatti, Bettina Frohnapfel, and Thomas Böhlke. Influence of flow–fiber coupling during mold-filling on the stress field in short-fiber reinforced composites. *Computational Mechanics*, 71(5):991–1013, March 2023.

[20] J.-F. Hétu, D. M. Gao, A. Garcia-Rejon, and G. Salloum. 3D finite element method for the simulation of the filling stage in injection molding. *Polymer Engineering & Science*, 38(2):223–236, 1998.

[21] Rong-Yeu Chang and Wen-Hsien Yang. Numerical simulation of mold filling in injection molding using a three-dimensional finite volume approach. *International Journal for Numerical Methods in Fluids*, 37(2):125–148, 2001.

[22] Sang-Woo Kim and Lih-Sheng Turng. Developments of three-dimensional computer-aided engineering simulation for injection moulding. *Modelling and Simulation in Materials Science and Engineering*, 12(3):S151, apr 2004.

[23] FOAM-extend project. `https://sourceforge.net/projects/foam-extend/`. Accessed: 2024-10-28.

[24] Bruno Ramoa, Ricardo Costa, Francisco Chinesta, and J. Miguel Nóbrega. A Semi-Automatic Approach Based on the Method of Manufactured Solutions to Assess the Convergence Order in OpenFOAM. *OpenFOAM® Journal*, 2:148–165, Nov. 2022.

[25] Saad M S Mukras and Fahad A Al-Mufadi. Simulation of HDPE mold filling in the injection molding process with comparison to experiments. *Arab. J. Sci. Eng.*, 41(5):1847–1856, May 2016.

[26] Markus Baum, Fabian Jasser, Michael Stricker, Denis Anders, and Simone Lake. Numerical simulation of the mold filling process and its experimental validation. *Int. J. Adv. Manuf. Technol.*, 120(5-6):3065–3076, May 2022.

[27] Henning Scheufler and Johan Roenby. TwoPhaseFlow: A Framework for Developing Two Phase Flow Solvers in OpenFOAM. *OpenFOAM® Journal*, 3:200–224, Dec. 2023.

[28] Xin Zhuang, Jie Ouyang, Chuntao Jiang, and Qingsheng Liu. New approach to develop a 3D non-isothermal computational framework for injection molding process based on level set method. *Chinese Journal of Chemical Engineering*, 24(7):832–842, 2016.

[29] Wei Wang, Xikui Li, and Xianhong Han. Numerical simulation and experimental verification of the filling stage in injection molding. *Polymer Engineering & Science*, 52(1):42–51, 2012.

[30] Lin Deng, Junjie Liang, Yun Zhang, Huamin Zhou, and Zhigao Huang. Efficient numerical simulation of injection mold filling with the lattice Boltzmann method. *Eng. Comput. (Swansea)*, 34(2):307–329, April 2017.

[31] William Han, Quentin Govignon, Arthur Cantarel, and Fabrice Schmidt. Efficient polymerization and crystallization kinetics coupling of polyamide 6 synthesis for liquid composite molding process modeling. *Polymer Engineering & Science*, 62(4):999–1012, 2022.

[32] V. Karyofylli, M. Behr, M. Schmitz, and C. Hopmann. Novel discretization methods for improved simulation precision in injection molding. *Materialwissenschaft und Werkstofftechnik*, 48(12):1264–1269, 2017.

[33] G. A. A. V. Haagh, G. W. M. Peters, F. N. Van De Vosse, and H. E. H. Meijer. A 3-D finite element model for gas-assisted injection molding: Simulations and experiments. *Polymer Engineering & Science*, 41(3):449–465, 2001.

[34] Wen Zhou Xuejuan Li, Jie Ouyang. Simulations of Three-dimensional Thermal Residual Stress and Warpage in Injection Molding. *Computer Modeling in Engineering & Sciences*, 96(6):379–407, 2013.

[35] M S Rusdi, M Z Abdullah, A S Mahmud, C Y Khor, M S Abdul Aziz, Z M Ariff, and M K Abdullah. Numerical investigation on the effect of pressure and temperature on the melt filling during injection molding process. *Arab. J. Sci. Eng.*, 41(5):1907–1919, May 2016.

[36] Jean-François Agassant, Pierre Avenas, Pierre J. Carreau, Bruno Vergnes, and Michel Vincent. *Injection Molding*, chapter 7, pages 521–585. Carl Hanser Verlag GmbH & Company KG, 2017.

[37] G. A. A. V. Haagh and F. N. Van De Vosse. Simulation of three-dimensional polymer mould filling processes using a pseudo-concentration method. *International Journal for Numerical Methods in Fluids*, 28(9):1355–1369, 1998.

[38] A. Polynkin, J. F. T. Pittman, and J. Sienz. 3D Simulation of Gas Assisted Injection Molding Analysis of Primary and Secondary Gas Penetration and Comparison with Experimental Results. *International Polymer Processing*, 20(2):191–201, 2005.

[39] J.U Brackbill, D.B Kothe, and C Zemach. A continuum method for modeling surface tension. *Journal of Computational Physics*, 100(2):335–354, 1992.

[40] Joseph P. Greene and James O. Wilkes. Numerical analysis of injection molding of glass fiber reinforced thermoplastics. Part 1: Injection pressures and flow. *Polymer Engineering & Science*, 37(3):590–602, 1997.

[41] C. Mnekbi, M. Vincent, and J. F. Agassant. Polymer rheology at high shear rate for microinjection moulding. *International Journal of Material Forming*, 3(S1):539–542, April 2010.

[42] D. Drummer and S. Messingschlager. Ceramic injection molding material analysis, modeling and injection molding simulation. *AIP Conference Proceedings*, 1593(1):582–586, 05 2014.

[43] Paul Zoller and Youssef A. Fakhreddine. Pressure—volume—temperature studies of semi-crystalline polymers. *Thermochimica Acta*, 238:397–415, 1994.

[44] Y. X. Yi and P. Zoller. An experimental and theoretical study of the PVT equation of state of butadiene and isoprene elastomers to 200°C and 200 MPa. *Journal of Polymer Science Part B: Polymer Physics*, 31(7):779–788, JUN 1993.

[45] Binxin Yang, Jie Ouyang, Chuntai Liu, and Qiang Li. Simulation of Non-isothermal Injection Molding for a Non-Newtonian Fluid by Level Set Method. *Chinese Journal of Chemical Engineering*, 18(4):600–608, 2010.

[46] Qingsheng Liu, Youqiong Liu, Chuntao Jiang, and Supei Zheng. Modeling and simulation of weld line location and properties during injection molding based on viscoelastic constitutive equation. *Rheol. Acta*, 59(2):109–121, February 2020.

[47] S.W. Chau and Y.W. Lin. Three-dimensional Simulation of Melt Filling and Gas Penetration in Gas-Assisted Injection Molding Process Using a Finite Volume Formulation. *Journal of Polymer Engineering*, 26(5):431–450, 2006.

[48] M. Tutar and A. Karakus. Injection Molding Simulation of a Compressible Polymer. *Journal of Polymer Engineering*, 29(6):355–384, 2009.

[49] Zixiang Hu, Yun Zhang, Junjie Liang, Songxin Shi, and Huamin Zhou. An efficient parallel algebraic multigrid method for 3D injection moulding simulation based on finite volume method. *International Journal of Computational Fluid Dynamics*, 28(6-10):316–328, 2014.

[50] Yun Zhang, Wenjie Yu, Junjie Liang, Jianlin Lang, and Dequn Li. Three-dimensional numerical simulation for plastic injection-compression molding. *Front. Mech. Eng.*, 13(1):74–84, March 2018.

[51] Suraj S Deshpande, Lakshman Anumolu, and Mario F Trujillo. Evaluating the performance of the two-phase flow solver interFoam. *Comput. Sci. Discov.*, 5(1):014016, November 2012.

[52] Tomislav Marić, Douglas B. Kothe, and Dieter Bothe. Unstructured un-split geometrical Volume-of-Fluid methods – A review. *Journal of Computational Physics*, 420:109695, 2020.

[53] Shaozhen Hua. Experimental and numerical investigation of jetting phenomenon in injection molding. *J. Braz. Soc. Mech. Sci. Eng.*, 42(4), April 2020.

[54] Li-Bo Chen, Yan-Hao Huang, Chuan-Yun Dai, Li-Qing Zhu, Xin Zhao, Kai Zhang, Xiao-Rong Fu, Wei Yang, and Ming-Bo Yang. Effects of convective schemes and geometric reconstruction scheme on interface of multiple melt flows. *Polymer*, 245:124692, 2022.

[55] OpenFOAM®. `https://www.openfoam.com/`. Accessed: 2024-10-28.

[56] Dongjin Seo, Jae Ryoun Youn, and Charles L. Tucker III. Numerical simulation of mold filling in foam reaction injection molding. *International Journal for Numerical Methods in Fluids*, 42(10):1105–1134, 2003.

[57] B. J. Araújo, J. C. F. Teixeira, A. M. Cunha, and C. P. T. Groth. Parallel three-dimensional simulation of the injection molding process. *International Journal for Numerical Methods in Fluids*, 59(7):801–815, 2009.

[58] M.G.H.M. Baltussen, M.A. Hulsen, and G.W.M. Peters. Numerical simulation of the fountain flow instability in injection molding. *Journal of Non-Newtonian Fluid Mechanics*, 165(11):631–640, 2010.

[59] S. R. Han, J. R. Cho, S. K. Beak, J. A. Hong, and Y. S. Lee. Numerical and experimental studies of injection compression molding process for thick plastic gas valve stem. *The International Journal of Advanced Manufacturing Technology*, 89(1–4):651–660, July 2016.

[60] Junjie Liang, Wan Luo, Zhigao Huang, Huamin Zhou, Yun Zhang, Yi Zhang, and Yang Fu. A robust finite volume method for three-dimensional filling simulation of plastic injection molding. *Engineering Computations*, 34(3):814–831, 2017.

[61] Lin Deng, Suo Fan, Yun Zhang, Zhigao Huang, Shaofei Jiang, Jiquan Li, and Huamin Zhou. A Novel Multiscale Methodology for Simulating Droplet Morphology Evolution during Injection Molding of Polymer Blends. *Polymers*, 13(1), 2021.

[62] P M Gresho. Incompressible Fluid Dynamics: Some Fundamental Formulation Issues. *Annual Review of Fluid Mechanics*, 23(Volume 23, 1991):413–453, 1991.

[63] P. Cardiff, Ž. Tuković, H. Jasak, and A. Ivanković. A block-coupled Finite Volume methodology for linear elasticity and unstructured meshes. *Computers & Structures*, 175:100–122, 2016.

[64] M. Tutar and A. Karakus. Computational Study of the Effect of Governing Parameters on a Polymer Injection Molding Process for Single-Cavity and Multicavity Mold Systems. *Journal of Manufacturing Science and Engineering*, 132(1):011001, 12 2009.

[65] Mixed boundary condition in OpenFOAM®. https://www.openfoam.com/documentation/guides/v2112/doc/guide-bcs-mixed.html. Accessed: 2024-10-28.

[66] Lennart Veltmaat, Felix Mehrens, Hans-Josef Endres, Jörg Kuhnert, and Pratik Suchde. Mesh-free simulations of injection molding processes. *Physics of Fluids*, 34(3):033102, 03 2022.

[67] Hrvoje Jasak and Tessa Uroić. *Practical Computational Fluid Dynamics with the Finite Volume Method*, pages 103–161. Springer International Publishing, Cham, 2020.

[68] S.V Patankar and D.B Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10):1787–1806, 1972.

[69] R.I Issa, B Ahmadi-Befrui, K.R Beshay, and A.D Gosman. Solution of the implicitly discretised reacting flow equations by operator-splitting. *Journal of Computational Physics*, 93(2):388–410, 1991.

[70] PIMPLE algorithm. https://doc.cfd.direct/notes/cfd-general-principles/the-pimple-algorithm. Accessed: 2026-01-13.

[71] F. Pimenta and M.A. Alves. Stabilization of an open-source finite-volume solver for viscoelastic fluid flows. *Journal of Non-Newtonian Fluid Mechanics*, 239:85–104, 2017.

[72] S.T. Miller, H. Jasak, D.A. Boger, E.G. Paterson, and A. Nedungadi. A pressure-based, compressible, two-phase flow finite volume method for underwater explosions. *Computers & Fluids*, 87:132–143, 2013. USNCCM Moving Boundaries.

[73] OpenFOAM® documentation on compressibleInterFoam. https://www.openfoam.com/documentation/guides/v2206/api/compressibleInterFoam_8C_source.html. Accessed: 2026-01-13.

[74] F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM and Matlab*. Springer Publishing Company, Incorporated, 1st edition, 2015.

[75] Patrick J. Roache. Code Verification by the Method of Manufactured Solutions. *Journal of Fluids Engineering*, 124(1):4–10, 11 2001.

[76] Bruno Blais and François Bertrand. On the use of the method of manufactured solutions for the verification of CFD codes for the volume-averaged Navier–Stokes equations. *Computers & Fluids*, 114:121–129, 2015.

[77] C. J. Roy, C. C. Nelson, T. M. Smith, and C. C. Ober. Verification of Euler/Navier–Stokes codes using the method of manufactured solutions. *International Journal for Numerical Methods in Fluids*, 44(6):599–620, 2004.

[78] Aniruddha Choudhary, Christopher J. Roy, Jean-François Dietiker, Mehrdad Shahnam, Rahul Garg, and Jordan Musser. Code verification for multiphase flows using the method of manufactured solutions. *International Journal of Multiphase Flow*, 80:150–163, 2016.

[79] Abouelmagd Abdelsamie, Ghislain Lartigue, Christos E. Frouzakis, and Dominique Thévenin. The Taylor–Green vortex as a benchmark for high-fidelity combustion simulations using low-Mach solvers. *Computers & Fluids*, 223:104935, 2021.

[80] Ricardo Costa, Stéphane Clain, Gaspar J. Machado, and João M. Nóbrega. Very high-order accurate finite volume scheme for the steady-state incompressible Navier–Stokes equations with polygonal meshes on arbitrary curved boundaries. *Computer Methods in Applied Mechanics and Engineering*, 396:115064, 2022.

[81] A. Veeraragavan, J. Beri, and R.J. Gollan. Use of the method of manufactured solutions for the verification of conjugate heat transfer solvers. *Journal of Computational Physics*, 307:308–320, 2016.

[82] Johan Roenby, Henrik Bredmose, and Hrvoje Jasak. *IsoAdvector: Geometric VOF on General Meshes*, pages 281–296. Springer International Publishing, Cham, 2019.

[83] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, September 2009.

[84] Peter H. Foss, Huan-Chang Tseng, John Snawerdt, Yuan-Jung Chang, Wen-Hsien Yang, and Chia-Hsiang Hsu. Prediction of fiber orientation distribution in injection molded parts using Moldex3D simulation. *Polymer Composites*, 35(4):671–680, 2014.

[85] cfMesh. `https://cfmesh.com/cfmesh-open-source/`. Accessed: 2024-10-28.

[86] Enhanced Structure CAE Solution with Molding Effect for Automotive Parts. `https://www.moldex3d.com/assets/2011/09/Enhanced-Structure-CAE-Solution-with-Molding-Effect-for-Automotive-Parts.pdf`. Accessed: 2024-10-28.

[87] Du Hwan Chung and Tai Hun Kwon. Invariant-based optimal fitting closure approximation for the numerical prediction of flow-induced fiber orientation. *Journal of Rheology*, 46(1):169–194, 01 2002.

[88] Charles L. Tucker. *Fundamentals of Fiber Orientation: Description, Measurement and Prediction*. Hanser Publishers, 2022.

Chapter 9 Conclusion and Future work

This work closes with a synthesis of the main findings and an outline of future research directions.

Each chapter was written based a manuscript prepared for scientific publication, therefore including its own conclusion section. In this chapter, the key results consolidated into an overarching conclusion, and relevant next steps are proposed.

9.1 General Conclusion

The research presented in this work enabled the application of emerging machine learning technologies to orientation modeling.

Chapter 2 demonstrated the use of a neural ODE to infer a closure relation from data generated in the dilute regime. To properly infer a closure model, admissible bounds were determined to maintain the predicted fourth-order tensor within physically admissible limits, considering orthotropic symmetry. Without these constraints, training the model proved extremely difficult.

When moving towards a concentrated orientation regime, Chapter 3 demonstrates that the orthotropic assumption should to be relaxed for certain flows. This is particularly evident in simple-shear, where non-orthotropic components emerge during the evolution of the orientation dynamics. Neglecting these components hampers key orientation features, leading to an erroneous trajectory. Accordingly, by relaxing orthotropy and adopting a monoclinic symmetry, empirical limits were established to keep the non-orthotropic components within a physically admissible range.

Since tensor-based orientation models require a closure relation, an attempt to learn new orientation indices was performed in Chapter 4, using planar configurations in steady-state. The information contained in a rich meso-scale orientation descriptor was effectively captured and compressed into a latent-space by a rank-reduction autoencoder technique. This reduced representation was guided by the reconstruction of a rich orientation descriptor, the second-order orientation tensor, and the suspension stress. A comparison with state-of-the-art orientation models demonstrated that the inferred model yielded better average predictions.

To democratize research on injection molding, including scenarios where fiber reinforced thermoplastic materials are required, a framework for modeling the process was created within the OpenFOAM® environment.

To emphasize the verification procedure for numerical tools, Chapter 5 presents the integration of the method of manufactured solutions into the OpenFOAM® framework in a semi-automatic manner, creating a simple workflow for solver and model verification. This procedure makes use of a high-level programming language (Python/SymPy) for symbolic computations and computes the required source terms, boundary conditions, and error metrics based on the desired manufactured solution. SymPy has built-in functionality to convert the symbolic expressions into C++ code that can be easily integrated into OpenFOAM®.

To enable the use of state-of-the-art fiber orientation models, given the lack of native numerical structures for storing fourth-order tensors in OpenFOAM® and corresponding arithmetic operations, a methodology similar to the one presented in Chapter 5, using symbolic computation and conversion of code to the OpenFOAM® format was used. To be able to solve the orientation dynamics, the second-order tensor resulting from the contraction of the fourth-order orientation tensor with the rate-of-deformation tensor is computed in SymPy, and the resulting symbolic expressions are converted into OpenFOAM® format. This procedure has the advantage of retaining only the non-trivial numerical operations, with the related algebraic operations being easier to implement in SymPy than to implement from scratch in OpenFOAM®. However, a major caveat of this approach is the loss of readability of the underlying mathematical operations. Chapter 6 illustrates this approach, and to enhance its utility for the community, it was implemented as a plug-in for incompressible solvers, allowing users to enable fiber-orientation analysis, without modifying their solver's underlying code.

The general framework for modeling the injection molding process is described in Chapter 7. The developed application includes multiple VoF algorithms, solution strategies, and boundary conditions. Several test cases are presented to demonstrate the verification procedure, together with a comparison against a well established commercial numerical tool. The results show the potential of the developed application, providing a robust, verifiable and easily extendable open-source platform for modeling the injection molding process.

9.2 Future work

The developed work advances the state of the art in orientation modeling by improving the understanding of the physical admissibility of orientation macro-descriptors and demonstrating the potential of artificial techniques for modeling orientation. Several straightforward extensions could broaden the reach of the developed techniques. The neural ODE closure can be trained to infer a closure relation using data in a concentrated regime. In fact, this direction was explored during this work; however, training was prohibitively slow due to the volume of data used, and it was therefore not pursued further. A key consideration for future development is the need for a stabilization technique during training to prevent system divergence. Even with empirically determined admissible bounds enforced, additional stabilization techniques appear to be required to improve training stability. Another straightforward extension, is the expansion of the Rank-Reduction Autoencoder methodology

to three-dimensional flows by using suitable machine learning architectures, such as convolutional networks. If successful, the dynamics of the reduced representation in latent space would also need to be determined.

Finally, during the development of this work, several algorithms were implemented, and attempts were made to recreate several works from the literature. However, the observed lack of information on reproducibility in some works led to the creation of an open-source tool to democratize injection molding research. The developed framework still needs several improvements, but two areas are particularly challenging: (i) adequate filling dynamics, either through the use of permeable boundary conditions or an alternative approach. If heat transfer is involved, care must be taken to ensure that energy is not artificially generated at the permeable boundaries; (ii) Computational speed-up. Within the work done several algorithms reported in the literature, with different pressure-velocity algorithms, were implemented and tested. An attempt with a block-coupled solver has been explored, since there is a report in the literature that claims a substantial speed-up. However, our implementation led to contrary observations. Because a successful block-coupled formulation was not achieved, future work could also encompass a detailed verification of the suitability of the block-coupled pressure-velocity strategy for solving transient incompressible problems, where larger time-steps are required to offset the computational demands of solving the linear system.