Lisa Maria König, BSc

Particle Separation in a Virtual Microfluidic Channel

MASTER'S THESIS

To achieve the university degree of

Diplom-Ingenieurin

Master's degree program: Chemical and Process Engineering

submitted to

Graz University of Technology

Supervisor

Ass.Prof. Dipl.-Ing. Dr.techn. Stefan Radl

Institute of Process and Particle Engineering

Graz, November 2015

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis dissertation.

Date

Signature

Abstract

In the paper industry the length and the shape of fibres are, among others, crucial for the quality of the final product. To be able to meet the high standards of paper products, the ability to separate unwanted fines from the pulp is of high importance. Furthermore, a fractionation of pulp into multiple classes, with different mean fibre lengths, would open the door for new, or improved fibre-based products, and more efficient processes. Unfortunately, already established separation or fractionation processes are often energy consuming, and/or require the addition of chemicals. Thus, the development of a new energy efficient and chemical-free concept to separate fines from fibre suspensions is desirable.

This thesis investigates, via computer simulation, the question whether so-called hydrodynamic fractionation devices are a feasible alternative for fibre fractionation. Therefore, fibre motion in a straight channel flow, the hydrodynamic fractionation of fibres at a T-junction, as well as the generation of fibre flocks are studied via detailed Euler-Lagrange simulations. Optimal parameters for an effective separation of fines (i.e., particles with a length smaller than 1mm) from a dilute pulp suspension are identified for the T-junction geometry. Furthermore, fibre flocks with a consistency up to 1%, with and without preferential fibre orientation were generated. These fibre flocks are now available for future simulations that may study the effect of fibre network formation on the separation efficiency.

Kurzfassung

In der Papierindustrie sind mitunter die Länge und Form der verwendeten Zellulose- und Holzstofffasern ausschlaggebend für die Qualität des Endproduktes. Um den immer vielfältigeren Anforderungen an die Papierprodukte gerecht zu werden, gilt es bereits die Zusammensetzung des Papierausgangsmaterials, zum Beispiel durch die Abtrennung von Feinstoffen, gezielt zu beeinflussen. Gegenwärtig etablierte Trennverfahren sind energieaufwändig und/oder verwenden Chemikalien, sodass die Entwicklung eines neuen energieeffizienteren, chemikalienfreien Konzeptes zur Abtrennung von Feinstoffen aus einer Fasersuspension erstrebenswert ist.

In dieser Arbeit wird durch Computersimulationen die Fragestellung untersucht, ob hydrodynamische Fraktionierung von Fasersuspensionen als Trennverfahren von Feinstoffen in der Papierindustrie eine interessante Alternative darstellt. Hierfür wird das Verhalten von Fasern in einer Rohrströmung, sowie die hydrodynamische Fraktionierung von Fasern und die Bildung von Faserflocken mittels CFD-DEM Simulationen beleuchtet.

Mit den verwendeten Modellen werden Parametersätze gefunden, welche das Abtrennen von Feinstoffen (d.h. Partikeln mit einer Länge <1mm) aus einer dünnen Suspension in Kanälen mit Abzweigern ermöglichen. Weiters können Faserflocken mit Konsistenzen bis zu 1%, mit zufälliger oder bevorzugter Faserausrichtung, hergestellt werden. Diese Flocken können in zukünftigen Simulationen zur Untersuchung des Einflusses von Netzwerkbildung auf die Trennwirkung eingesetzt werden.

Acknowledgement

This master thesis was done in 2015 at the Institute of Process and Particle Engineering at TU Graz, Austria. Here I want to thank all the people who made this thesis possible.

I owe my deepest gratitude to my supervisor Ass.Prof. Dipl.-Ing. Dr.techn. Stefan Radl. Without his continuous support, engagement and dedication this work would have not been possible. I thank him for his great effort he put into guiding me into the scientific field of simulation, and for his flawless editing of this work.

I would like to thank all members of the cooperative research project *FLIPPR* who enabled this thesis.

I am grateful to the IT Services from the Graz University of Technology, especially I want to thank the ZID for providing the necessary computational power on the *dCluster* to enable my simulation runs for this thesis.

I am indebted to many of my colleagues who supported me throughout this thesis. Especially I want to thank Dipl.-Ing. Jakob Redlinger-Pohn, BSc and Josef König, BSc , they have made their support available in a number of ways.

My sincere thanks go to my colleagues and friends I was lucky to find in the Chemical and Process Engineering – "family" at the TU Graz, in particular Matthias Grabner, BSc, who always helped me with words and deeds throughout my whole study.

It is a pleasure for me to thank my parents for their continuous support – spiritually and materially. Without them I had never been able to finish this thesis and more my whole Chemical and Process Engineering studies. Thank you for giving me the opportunity to gain this education.

I am thankful to my sister who introduced me to the exciting field of Chemical and Process Engineering.

Last but not least I want to express my profound gratitude to my partner Julia, who always tried to follow my technical excursions and supported me in so many ways.

Table of Content

1 Introduction 1

1.1 The Importance of Process Virtualization 1

1.2 Fibre-Fines Separation in the Pulp and Paper Industry 2

1.3 Inertial Microfluidics 2

1.4 Hydrodynamic Fractionation 3

1.5 Goals 4

2 State of the Art 5

2.1 CFD Basics 5

2.1.1 Inlet Conditions 5

2.2 DEM Basics 6

2.2.1 Non-spherical Particles 6

2.3 Coupling 7

3 Simulation Strategy 9

3.1 Fluid Flow near a T-Junction 9

3.1.1 T- Junction Geometry 9

3.1.2 Meshing 10

3.1.3 Numerical Schemes 11

3.1.4 Boundary Conditions 12

3.2 Particle Motion 15

3.2.1 Fibre Model 15

3.2.2 Fibre-Fibre and Fibre-Wall Interactions 15

3.2.3 Fibre-Fluid Interactions 16

3.2.4 Coordinate System 17

3.2.5 Benchmark Simulation: Fibre Flock Formation 19

4 Results for Fluid Flow 25

4.1 Volumetric Flow Fraction exiting through the Side Channel 25

4.2 Pressure Distribution near a T-Junction 26

4.3 Predicting Flow Rates through a T-Junction 28

5 Fibre-Fines Separation in a Dilute Suspension 32

5.1 Explorative Studies 32

5.1.1 Fiber Behavior in a Straight Channel under Laminar Flow Conditions 32

5.1.2 Particle-Wall Impact 34

5.1.3 Setup for the Analysis near a T-Junction 35

5.1.4 Critical Separation Layer 36

5.1.5 Impact of the Side Channel on Unseparated Fibres 40

5.2 Full Simulation Setup 41

5.2.1 Fibre Position and Orientation at the Inlet 41

5.2.2 Flow Setup 41

5.3 Results for Fibre-Fines Separation near a T-Junction 43

5.3.1 Separation Efficiency 43

5.4 Fibre Flock Behavior near a T-Junction 50

6 Mechanism behind Fibre-Fines Separation 51

7 Conclusions 52

7.1 Key Achievements 52

7.2 Comparison with other Separation Technologies 53

7.3 Limitations and Outlook 54

8 References 56

Appendix A CFD Simulation 62

A.1 Straight Channel 62

A.2 Discretization Schemes [38] 63

A.3 Pressure Loss 63

A.4 Volumetric Flow Fraction 64

A.4.1 Relationship between Pressure and Volumetric Flow Fraction 64

A.5 Dimensionless Wall Distance y^+ 66

A.6 CFD Benchmark Simulation Data 67

A.6.1 Detailed Case Data 67

A.6.2 Laminar Inlet Profile 69

A.6.3 Mapped Inlet Profile 69

A.6.4 Comparison of Laminar and Mapped Inlet Profiles 73

A.6.5 Steady State Case SCW1.0_90_20_10 75

Appendix B Fibre Motion 77

B.1 Eccentricity of Prolate Spheroids 77

B.2 Hydrodynamic Force F^h and Torque T^h 78

B.2.1 Viscous Effects 78

B.3 Contact Forces F^c and Torque T^c 78

B.3.1 Contact Forces in the Roughness Layer Region 79

B.3.2 Contact Forces in the Fibre Contact Region 80

B.4 Lubrication Force 81

B.5 Contact Point Detection 82

B.5.1 Line-Line Distance 82

B.5.2 Line-Triangle Distance 84

Appendix C DEM Simulation 87

C.1 Fibre-Fibre Interaction Test Case 87

C.1.1 Collision Point and Contact Vector 87

C.1.2 Post-Collisional Translational and Angular Velocities 89

C.1.3 Coefficient of Restitution 91

C.1.4 Energy Conservation 92

C.1.5 Momentum Conservation 92

C.1.6 Test Case Results 93

C.2 Fibre-Flock Generation Pre-Processing 95

C.2.1 LIGGGHTS® input 95

C.2.2 General Definitions 95

C.2.3 Pre-Calculations 96

C.2.4 Number of Fibres in each Class *i* 97

C.2.5 Initial Fibre Position and Orientation 97

C.2.6 Time Step Parameters 99

C.2.7 Deformation Rate 99

C.2.8 Fibre Overlap 99

C.2.9 Trate Input Parameter R [56] 99

C.2.10 Run Input Parameter 100

C.3 Fibre-Flock Generation Post Processing 101

C.3.1 LIGGGHTS® Output Data 101

C.3.2 Dividing Fibres into the 4 Sections according to their Position 101

C.3.3 Fibre Orientation 102

C.3.4 Angular Velocity 102

C.3.5 Mathematical Basics 105

Appendix D CFD-DEM Simulation 107

D.1 Jeffery Orbit 107

D.2 Velocity Profile in a Square Cross Section 107

D.3 Shear Rate 107

D.4 Half Orbit Time and Traveling Length 109

D.5 Fibre Orbit: Analytical vs. Numerical Results 109

Appendix E Dimensionless Numbers 111

Appendix F MATLAB Code 112

List of Figures

Fig. 1: Principle of hydrodynamic fractionation at a T-junction consisting of a main and a perpendicular side channel. Particles are separated according to their size. Small particles leave the main channel through the side channel, large particles stay in the main channel [19]. 3

Fig. 2: T-junction geometry. Left panel: front view of the T-junction. Right panel: side view of the T-junction. Grey lines: qualitative representation of the mesh cells‘ size and shape. 9

Fig. 3: Unsymmetrical fluid flow profile when using a refined mesh. Time-averaged velocity in main channel x^+-direction, shortly after the junction in the T-junction (Re = 5,000, Δp^+ = -0.30)........ 10

Fig. 4: Laminar inlet profile. Left panel: isometric view. Right panel: y^+z^+-plane of main channel inlet. 13

Fig. 5: Internal mapping to generate the inlet flow profile. The grey-shaded plane represents the cross section which is mapped back to the inlet........ 14

Fig. 6: Fibre geometry assumptions. Top panel: spherocylinder. Bottom panel: prolate spheroid. 15

Fig. 7: Fibre with roughness layer to represent fibrils. 16

Fig. 8: Coordinate systems used in this study. Left panel: global coordinate system. Right panel: local cross sectional coordinate system, as well as fibre coordinate system........ 17

Fig. 9: Simulation box, divided into 4 equally sized triangular sections 1, 2, 3 and 4. Each section i has its own section-based coordinate system B_i'' ($e_{x,i}''$, $e_{y,i}''$, $e_{z,i}''$). Middle: Global coordinate system B (e_x, e_y, e_z). 18

Fig. 10: Spherical coordinate system [41]........ 18

Fig. 11: Three possible extreme fibre orientations in each sub-section of the cross section. .. 19

Fig. 12: Illustration of the compaction procedure to produce a fibre flock. Left panel: tri-axial deformation. Right panel: uni-axial deformation in x^+-direction........ 21

Fig. 13: Dimensions of the box used for the compaction procedure. Left: original box. Middle: final (post-deformation) box and reduced final (reduced post-deformation) box. Right: detail: fibre in the corner of the reduced final box........ 21

Fig. 14: Snapshots of the fibre flocks. Top: fibre flock generation with tri-axial deformation method. Bottom: fibre flock generation with uni-axial deformation method. Left: total fibre flock. Right: fibre flock detail........ 23

Fig. 15: Initial orientation distribution for the tri- and uni-axial deformation process. 24

Fig. 16: Tri-axial deformation: fibre orientation at the end of the simulation, φ_{Fib} =1.0%...... 24

Fig. 17: Uni-axial deformation: fibre orientation at the end of the simulation, φ_{Fib} =1.0%..... 24

Fig. 18: Volumetric flow fraction leaving the T-junction via the side channel. Re = 500, $L_{outlet}^{+} = L_{side}^{+} = 6$. 25

Fig. 19: Pressure distribution at a T-junction (L_{inlet}^{+}=20, L_{oulet}^{+}=L_{side}^{+}= 6, SCW^{+}=1, α=90°). Top penal: Δp^{+}=-0.30. Middle panel: Δp^{+}=0.00. Bottom panel: Δp^{+}=0.35. 26

Fig. 20: Pressure distribution in the side channel of a T-Junction (L_{inlet}^{+}=20, L_{oulet}^{+}=L_{side}^{+}= 6, SCW^{+}=1, α=90°). Left penal: Δp^{+}=-0.30. Center panel: Δp^{+}=0.00. Right panel: Δp^{+}=0.35... 27

Fig. 21: Overall pressure loss over a single T-junction (Reynolds number Re = 500, outlet length L_{outlet}^{+} = 6, inlet length L_{inlet}^{+} and side channel width SCW^{+} vary in these simulations; left panel: T-junction geometry with an inlet length of L_{inlet}^{+} = 20 (SCW^{+}=1) and L_{inlet}^{+}= 20.5 (SCW^{+}= 0.5); right panel: T-junction geometry with an inlet length of L_{inlet}^{+}= 40 (SCW^{+}=1) and L_{inlet}^{+}= 40.5 (SCW^{+}= 0.5)). 28

Fig. 22: Schematic representation of the pressure profiles near a T-junction. 28

Fig. 23: Optimal loss coefficient for the main channel for different channel configurations (L_{inlet}^{+}=20, L_{oulet}^{+}=L_{side}^{+}= 6). 30

Fig. 24: Predicted versus simulated volumetric flow fraction at a T-junction (ζ = 65.5, ζ_J as per Fig. 23, L_{inlet}^{+}=20, L_{oulet}^{+}=L_{side}^{+}= 6). 31

Fig. 25: Fibre repositioning due to a single fibre-wall interaction (flow is left to right). Blue line: initially positioned and oriented fibre. Grey lines(s): rotating fibre. Orange line: the fibre just contacts the wall. Red line: repositioned fibre located at a distance s-$d_{Major,AR}/2$ from the wall. The fibres center of mass moves Δz^{+} towards the center of the channel due to a fibre-wall interaction. 33

Fig. 26: Jeffery orbit calculations. Left Panel: time period of a half orbit $t^{+}_{0.5orbit}$. Right panel: traveling length of a half orbit $s^{+}_{0.5orbit}$. 34

Fig. 27: Schematic representation of the ideal fibre position and orientation setup in a straight channel to investigate the impact of a fibre-wall interaction. 34

Fig. 28: The effect of fibre-wall interaction: fibres reposition due to wall contact at a distance z^{+}=s-$d_{Major,AR}/2$, i.e., (s-|z^{+}_{end}|)/($d_{Major,AR}/2$)=1 (initial fibre position and orientation as shown in Fig. 27). 35

Fig. 29: Schematic representation of the data collection planes near the T-junction. 35

Fig. 30: Schematic representation of the initial fibre position and orientation in a T-junction to probe the position of the critical separation layer. Top: fibres oriented in x^{+}-direction. Bottom: fibres oriented in y^{+}-direction. 36

Fig. 31: Critical separation layer as a function of the fibres aspect ratio (left column: fibres initially oriented in the streamwise direction; right column: log-rolling fibres). 39

Fig. 32: Impact of the downwards suction of the side channel on the fibres remaining in the main channel. Left panel: fibres initially oriented in x^+-direction. Right panel: fibres initially oriented in y^+-direction. All fibres are initially positioned at $y^+ = 0$. The side channel entrance is located at $s+z_{preT}^+ = 0$, and the opposite wall is located at $s+z_{preT}^+ = 1$ (case: SCW^+=1, α=90°, Δp^+=0.35). 40

Fig. 33: Initial fibre positions in the y^+z^+-plane located at $x^+ = -20$ (for L_{inlet}^+=20) or $x^+ = -40$ (for L_{inlet}^+=40). Left panel: representation of each fibre's center of mass position. Right panel: fibres represented by cylinders in the channel. 41

Fig. 34: Dilute fibre suspension flowing in T-junction at different instances in time. Fibres in the center of the main channel travel faster, and pass the junction more often than fibres closer to the walls. 42

Fig. 35: Separation efficiency using different inlet lengths. Left panel: separation efficiency at first T-junction. Right panel: separation efficiency over whole simulation duration. 44

Fig. 36: Separation efficiency of the most promising parameter settings. 44

Fig. 37: Separation efficiency for best parameter settings. 45

Fig. 38: Dilute fibre suspension in a channel with a T-junction at optimal operating conditions (case: SCW0.5_45_dp0.25). Left panel: fibres at the beginning of the simulation at the inlet (t^+=1). Right panel: T-junction, fibres from class AR = 2 and AR = 5 close to the bottom wall leave through the side channel (t^+ = 116.5). 46

Fig. 39: Schematic representation of the simulation setup to study the extension of the critical separation layer. Fibres are positioned in a C-shaped region over the whole cross section, and are oriented in the x^+-direction. 47

Fig. 40: Critical separation lines in the y^+z^+-plane for all fibre classes. Fibres are initially placed in the streamwise direction, and are injected just before the side channel at x^+=-0.51 (case: SCW^+=1, α=90°, Δp^+=0.35). 47

Fig. 41: Orientation of all fibres at x^+=-0.51 (case: SCW^+=1, α=90°, Δp^+=0.35), all classes have the same width of 15 degrees. 48

Fig. 42: Geometry of a single baffle. 48

Fig. 43: Three shifted baffles in the inlet section of the channel. Top panel: side view of inlet section. Bottom panel: top view of inlet section. 49

Fig. 44: Influence of baffles on the separation efficiency (case: L_{inlet}^+=20, SCW^+=1, α=90°, Δp^+=0.35). Left panel: at first T-junction. Right panel: over the whole simulation time. 50

Fig. 45: Straight channel geometry. Grey mesh: quantitative representation of the mesh cells and their size........ 62
Fig. 46: Comparison of the pressure and the time-averaged pressure along a sample line. Top: sample line in y^+-direction. Bottom: sample line in z^+-direction (case: SCW1.0_90_20_10). 73
Fig. 47: Comparison of the velocities and the time-averaged velocities along a sample line. Top: sample line in y^+-direction. Bottom: sample line in z^+-direction (case: SCW1.0_90_20_10)........ 74
Fig. 48: Residuals over ~28·10^3 iterations........ 75
Fig. 49: Velocity and pressure at different probe positions in T-junction. 1st row: Velocity components in x^+-, y^+-, z^+- direction at the beginning and the end of the main channel. 2nd row: Velocity components in x^+-, y^+-, z^+- direction at the end of the side channel and right at the beginning of the junction. 3rd row: Velocity components in x^+-, y^+-, z^+- direction right at the end of the junction and the area-averaged pressure at the inlet of the main channel......... 76
Fig. 50: Schematic representation of linear spring-dashpot model used for fibre interaction. Graphic design taken from Goniva [57]........ 79
Fig. 51: Two approaching cylinders. Left panel: cylinders are non-parallel, angle α. Right panel: cylinders are parallel........ 81
Fig. 52: Distance between two line segments [30]........ 82
Fig. 53: Left panel: domain for *s* and *t*. Right panel: Domain region 0 for s and t and boundary domains region1-8 [30]........ 83
Fig. 54: Line-triangle distance determination [30]........ 84
Fig. 55: Line-triangle contact determination [30], [58]........ 84
Fig. 56: Three dimensional solution space for line-triangle distance determination [30]........ 85
Fig. 57: Fibre-fibre collision areas. Red line shows the collision area of fibre A (does change due to angle α). Red dot shows collision point on fibre B (does not change due to the angle α, but due to x)........ 88
Fig. 58: Results fibre-fibre interaction code testing: analytical vs. numerical solution. Top row: fibre 1 (=A). Left: angular velocity around y^+-axis. Right: velocity in x^+- and z^+-direction. Middle row: fibre 2 (=B). Left: angular velocity around y^+-axis. Right: velocity in x^+- and z^+-direction. Bottom row: Left: total energy of both fibres. Right: linear momentum of both fibres........ 94
Fig. 59: 2e4 uniformly distributed points on the surface of a unit sphere. All classes have the same width of 15 degree. Top: points on the surface of the sphere. Left: azimuthal angle distribution. Right: polar angle distribution........ 98

Fig. 60: Coordinate Transformation B_i'' → $B_{j,i}'$ 103
Fig. 61: Velocity and shear rate profile in channel with laminar flow field, fibre at r_{max}. 110
Fig. 62: Comparison of the numerical and the analytical solutions of a 180° spin of a fibre in a linear flow field. Blue curve: Rotational energy in the system. Red dot: first minima after 180° spin. Black dot: analytical dimensionless time result for a 180° spin. Left panel: shortest fibre *AR*=2. Right panel: longest fibre *AR*=20. 110

List of Tables

Table 1: Numerical schemes used in OpenFOAM® 12
Table 2: T-junction: fluid pressure boundary conditions. 12
Table 3: T-junction: fluid velocity boundary conditions. 13
Table 4: Analysis of the three possible extreme fibre orientations in terms of spherical coordinates. 19
Table 5: Fibre types: dimensions and distributions. 20
Table 6: Fibre flock generation setup for tri- and uni-axial deformation. 22
Table 7: Fibre flock data for tri- and uni-axial deformation. "real"-values represent properties of the final fibre flock 22
Table 8: Cases with the most promising parameter settings. 38
Table 9: Cases with the best separation efficiency results. 45
Table 10: Baffle dimensions. 49
Table 11: Comparison of hydrodynamic fractionation devices (HFD) versus pressure screens (PS, [12]). 54
Table 12: Straight channel: fluid pressure boundary conditions. 62
Table 13: Straight channel: fluid velocity boundary conditions. 62
Table 14: CFD Cases Simulation Data Part 1. 67
Table 15: CFD Cases Simulation Data Part 2. 68
Table 16: Test Case Parameters. 93
Table 17: Test Case min./max. contact time steps for fibre-fibre interaction. 94
Table 18: Fibre sectioning 101
Table 19: Change of basis B → B_i'': rotation angle for each section *I*, around the global x^+-axis. 102
Table 20: Jeffery orbit calculation. 110

Abbreviations

CFD	Computational Fluid Dynamics
CFDEM®	Open-source code that realizes coupling of CFD and DEM
DEM	Discrete Element Method
DNS	Direct Numerical Simulation
FDM	Finite Difference Method
FEM	Finite Element Method
FVM	Finite Volume Method
in_C	inlet plane boundary of the main channel
LES	Large Eddy Simulation
LIGGGHTS®	LAMMPS improved for general granular and granular heat transfer simulations
OpenFOAM®	(Open Source) Field Operation And Manipulation
out_C	outlet plane boundary of the main channel
out_J	outlet plane boundary of the side channel
post_T	y^+z^+ data collection plane shortly after side channel
pre_T	y^+z^+ data collection plane shortly before side channel
RANS	Reynolds Averaged Navier Stokes equations
SCS	square cross section
side_C	x^+y^+ data collection plane shortly after side channel entrance

Nomenclature

Latin Symbols

A	cross sectional area	m²
$\mathbf{cm_A}$	center of mass of fibre A	-
$\mathbf{cm_{B,\alpha}}$	center of mass of fibre B	-
D	charateristic particle diameter	m
d_i	length of a line segment	m
d_{Major}	long diameter of spheroid, spherocylinder	m
d_{Minor}	short diameter of spheroid, spherocylinder	m
$\mathbf{e_{nP\alpha}}$	unit vector normal to collision plane	-
H_{hyd}	hydraulic diameter	m
$\hat{I}$	momentum exchange due to a collision	kg m s^{-1}
k	spring stiffness	-
$\mathbf{L}$	angular momentum	kg m^2 s^{-1}
$\mathbf{L_i}$	line segment	-
L^+	dimensionless height and width of the main channel's cross section	-
L_{inlet}^+	dimensionless length of the main channel's inlet section	-
L_{side}^+	dimensionless length of the side channel	-
L_{outlet}^+	dimensionless length of the main channel's outlet section	-
m_1 m_2	mass of particle 1 / 2	kg
m_A m_B	mass of fibre A / B	kg
$\mathbf{M_{A1}}$	center point of fibre A’s right half-sphere (spherocylinder)	-
p	pressure	kg m^{-1} s^{-2}
$\mathbf{P}$	linear momentum	kg m s^{-1}
$\mathbf{P_i}$	basis point of a line segment	-
$P_{stat,in}$	static pressure at inlet of main channel	Pa
$P_{stat,out}$	static pressure at outlet of main channel	Pa
$\mathbf{P_\alpha}$	collision point of two fibres	-
$Q\phi$	sources and sinks	-
Re	Reynolds number	-
s	half height/width of the main channel's cross section	m
SCW^+	dimensionless side channel witdh	-
t	Time	s
t_c	contact time, typical response time	s
U	cross section circumference, characteristic fluid velocity	m / m s^{-1}

$\bar{u}$	spatially-averaged velocity	m s^{-1}
$<\bar{u}>$	time- and spatially averaged velocity	m s^{-1}
$<u>$	time-averaged velocity	m s^{-1}
$<\bar{u}_{inC}>$	time- and spatially averaged velocity at inlet of main channel	m s^{-1}
$<\bar{u}_{outC}>$	time- and spatially averaged velocity at outlet of main channel	m s^{-1}
$<\bar{u}_{outJ}>$	time- and spatially averaged velocity at outlet of side channel	m s^{-1}
u_τ	friction velocity	m s^{-1}
$\mathbf{v_{A1}}$	initial (pre-collision) velocity of fibre A's center of mass	m s^{-1}
$\mathbf{v_{A2}}$	final (post-collision) velocity of fibre A's center of mass	m s^{-1}
$\mathbf{v_{B1}}$	initial (pre-collision) velocity of fibre B's center of mass	m s^{-1}
$\mathbf{v_{B2}}$	final (post-collision) velocity of fibre B's center of mass	m s^{-1}
$\mathbf{v_{PA,1}}$	velocities of the collision point on fibre A before collision	m s^{-1}
$\mathbf{v_{PA,2}}$	velocities of the collision point on fibre A after collision	m s^{-1}
$\mathbf{v_{PB,1}}$	velocities of the collision point on fibre B before collision	m s^{-1}
$\mathbf{v_{PB,2}}$	velocities of the collision point on fibre B after collision	m s^{-1}
y^+	dimensionless wall distance	-

Greek Symbols

α	angle between main and side channel/angle between fibres' center line	° / rad
γ_0	viscous damping coefficient	-
Γ	diffusion coefficient	-
Δp^+	dimensionless pressure difference between main and side outlet	-
ΔP_{loss}	pressure loss in main channel	Pa
$\mathbf{\Delta v_A}$	change of velocity of fibre A	m s^{-1}
$\mathbf{\Delta v_B}$	change of velocity of fibre B	m s^{-1}
$\mathbf{\Delta \omega_A}$	change of angular velocity of fibre A	rad s^{-1}
$\mathbf{\Delta \omega_B}$	change of angular velocity of fibre B	rad s^{-1}
ε	coefficient of restitution	-
ζ	loss coefficient	-
η_0	rescaled viscous damping coefficient	-
θ	azimuthal angle	° /rad
μ	dynamic viscosity	m² s^{-1}
ν	kinematic viscosity	kg m^{-1} s^{-1}
ρ	density	kg m^{-3}
τ_w	wall shear stress	kg m^{-1} s^{-2}

φ	consistency	-
Φ^+	volumetric flow fraction	-
ϕ	polar angle, characteristic fluid parameter	° /rad /-
ω	eigenfrequency of a contact	s-1
$\boldsymbol{\omega}_{\mathbf{A1}}$	initial (pre-collision) angular velocity of fibre A	rad s^{-1}
$\boldsymbol{\omega}_{\mathbf{A2}}$	final (post- collision) angular velocity of fibre A	rad s^{-1}
$\boldsymbol{\omega}_{\mathbf{B1}}$	initial (pre- collision) angular velocity of fibre B	rad s^{-1}
$\boldsymbol{\omega}_{\mathbf{B2}}$	final (post collision) angular velocity of fibre B	rad s^{-1}

Indices

min	minimum
max	maximum
ave	averaged
c	main channel
cr	critical
p	particles
i	fibre index (e.g., A, B)
j	side channel
1	initial (pre-collision) value
2	final (post-collision) value
fluid	fluid
deform	deformation
tot	total

1 Introduction

1.1 The Importance of Process Virtualization

In 2011 the term *Industry 4.0* was officially used for the first time. *Industry 4.0* stands for a new High-Tech strategy to make *Smart Factories* possible, and initiate the forth industrial revolution. The goal of this strategy is the intelligent network of machines, work pieces, systems and employees to raise the way of production to another level, where optimized products are produced in an optimized environment. The impact of *Industry 4.0* is widely spread, e.g., it will improve machine safety, the industrial value chain, our socio-economics, and the conditions of work. In the sense of production it will be the next big step of optimization to efficiently produce individualized, high quality products with fewer resources in a shorter period of time. *Industry 4.0* is based on six design principles: one of them is virtualization with which reality is virtually reproduced to administrate, organize and optimize processes in advance, in the present and in the future ([1], [2]).

But not only existing processes, also new concepts and ideas can be virtually investigated by simulations for research and development purposes. Simulations offer a series of advantages over experiments. Simulations do not require a certain lab stand that needs to be built. Virtual experiments can be run in parallel, with multiple parameter settings which make simulations less time-consuming and less expensive. There is no human or environmental risk of carrying out experiments, even if hazardous conditions are studied. In simulations every detail can be looked at (theoretically), even in areas where in reality suitable measurement devices do not exist, or are expensive. Simulations can provide detailed insight into what is happing in a processes, and consequently help in the identification of correlations that describe a process [3]. Clearly, it can be expected that in the close future the dependency of simulations to be verified by experiments will disappear, leading to a fruitful symbiosis of simulation models and reality [4].

Despite great advances have been made in the recent past with respect to process virtualization, there are scientific disciplines where detailed process modeling is still rarely done. One example for such a discipline is pulp and paper processing: due to the nature of fibre suspensions, i.e., the flexibility, size and shape of suspended fibres, significantly more rigorous simulation models are required compared to, e.g., fluid flow processes. These facts lead to significantly more sophisticated models, longer simulation times, and ultimately to a higher cost of modeling. Next, we detail on a relevant process in the pulp and paper industry, i.e., fibre-fines separation, to illustrate challenges that arise when attempting to model such a process.

1.2 Fibre-Fines Separation in the Pulp and Paper Industry

Pulp, the feed material for paper, is a multicomponent suspension: it consists of fibres, i.e., elongated flexible particles with a length of > 200µm, and fines, i.e., comparably small particles with an aspect ratio close to one [5]. The properties of the pulp and the final paper product strongly depends on the concentration of fines in the pulp [6], e.g. fines increase the strength of the paper products [7], or lower the porosity of the paper. Depending on their size and shape, fines can satisfy structural functions such as filling gaps between long fibres [5], or increase the tensile strength, the density, and influence the optical properties [8].

To provide the optimal paper for a certain application, the fines concentration needs to be controlled. To realize this goal, a separation step for fibres and fines is needed. Commonly used equipment for fibre-fines separation in the pulp and paper industry are (i) pressure screens [9], [10], (ii) specialized washing units (typically focusing on de-ashing, e.g., [11]) or (iii) flotation devices. Unfortunately, flotation is not a chemical-free method, and pressure screens require significant energy input [12]. De-ashing washing units have a similar design as a headbox of a paper machine, and hence typically consume significant space and energy during operation [11]. Thus, a separation device using little energy and no chemicals is preferable. Hydrodynamic fractionation, i.e., a separation process that relies on flow phenomena in small channels, seems to be a promising starting point for the development of such a device.

1.3 Inertial Microfluidics

Inertial microfluidics deal with flow phenomena in narrow channels that are characterized by small, but finite Reynolds number flow. A flow regime is characterized by the fluid Reynolds number which represents the ratio of inertial and viscous forces: $Re = \rho UH/\mu$ [13]. In channel flow, for example, the critical Reynolds number is Re_{cr} = 2,300, indicating the starting point for the transition from laminar to turbulent flow [14]. In the early days of microfluidic systems it was common to incorrectly assume that for flow regimes with $Re \ll Re_{cr}$, inertial effects can be neglected. However, recent investigations have clearly shown that fluid inertia is the reason for useful effects that can be exploited in, e.g., particle separation devices. Even in a laminar flow regime characterized by $1 < Re < \sim 100$, effects due to the fluid's inertia can be observed, e.g., the effect of curved or bifurcated channels on the distribution of suspended particles [13].

In particle separation processes effects due to non-zero inertial forces, e.g., lift forces on particles forcing them to move perpendicular to the stream direction, can, if comprehended, be used to manipulate the flow of suspended particles [15]. Note, these inertial effects are fluid-related, and hence do not depend on the density difference between the fluid and the particles. Consequently, these effects can be utilized for the separation of particles that have a density close to that of the ambient fluid. Clearly, inertial microfluidics offers an interesting playground for fibre suspension separation, e.g., in hydrodynamic fractionation devices.

1.4 Hydrodynamic Fractionation

One way to use inertial effects for the separation of particles is hydrodynamic fractionation (HDF). In hydrodynamic fractionation devices there are no external forces or fields, i.e., it is a so-called passive separation. Only the fluid flow and the particle-interaction cause a separation of particles according to their shape and size [16]. There have been already some applications of HDF in the biomedical research field ([17], [18]), and in length-depending sorting processes of rod-like particles in the life sciences [19]. One well-investigated way to realize HDF for separating spherical particles according to their size is the usage of a series of T-junctions. A particle suspension flows through a main channel with multiple perpendicularly arranged side channels.

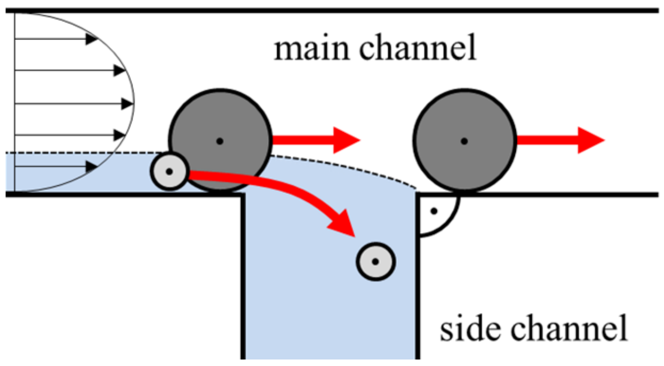


Fig. 1: Principle of hydrodynamic fractionation at a T-junction consisting of a main and a perpendicular side channel. Particles are separated according to their size. Small particles leave the main channel through the side channel, large particles stay in the main channel [19].

While the suspension flows through the apparatus, a small amount of liquid is removed from the main channel through side channels. The removed liquid drags smaller particles with it, whereas larger particles stay in the main channel [16]. In Fig. 1 the principle of the separation process is shown. The dashed line represents the critical separation line. The liquid and the particles beneath this critical separation line are removed from the main channel through the side channel. In contrast, the liquid and the particles with their center of mass above the critical separation line stay in the main channel [19]. Although HDF has been applied to particle suspensions involving spherical and moderate aspect ratio particles, there is a gap of understanding with respect to the applicability of HDF to fibre suspension separation and fractionation.

1.5 Goals

This master's thesis aims on contributing to the investigation of a new concept in fibre-fines separation using the principle of HDF. The main object is the understanding of the fibres' behavior in channel flow, and the description of the mechanism behind the hydrodynamic fractionation of a fibre-fines suspension by means of simulations.

The main steps in this thesis are:

- investigation of the fluid flow near a T-junction
- introduction of a stiff fibre model, accounting for fibre-wall and fibre-fibre interactions
- generation and investigation of the behavior of a fibre flock
- study of fibre behavior in a straight channel
- investigation of the separation process of a dilute fibre suspension near a T-junction
- determination of a parameter set that allows hydrodynamic fractionation of a fibre suspension

2 State of the Art

2.1 CFD Basics

Computational Fluid Dynamics, short CFD, became a powerful tool to investigate fluid flow during the last three decades. Nowadays, a variety of CFD models are available to predict real-world fluid flow at different levels of fidelity, e.g. DNS, LES, or RANS [20]. By using a DNS (short for Direct Numerical Simulation), the fluid flow is fully resolved and no additional modeling (except that inherent to the Navier-Stokes equations) is applied. Thus, the Navier-Stokes equations are directly solved, which provides the highest level of insight to fluid flow behavior, e.g. in different geometries [21]. A well-established tool for DNS simulations in scientific research is the open source software package OpenFOAM®. The suitability of the schemes available in OpenFOAM® for DNS is good, although limited to a 2nd order accuracy [22].

For the investigation of fluid flow behavior, the flow regime is one of the most important factors. The flow regime in a channel depends on the Reynolds number *Re*. If the Reynolds number exceeds the critical Reynolds number $Re_{cr} \approx 2{,}300$*,* the flow regime transitions from laminar flow to turbulent flow [14]. The flow regime is not only influenced by the Reynolds number: another crucial factor is the flow profile at the inlet, since the inlet conditions can have a significant impact on the fluid behavior throughout the whole simulation domain. Unfortunately, the treatment of the inlet condition is quite challenging, especially for non-laminar flows [23]. Hence possible inlet profile procedures are described next.

2.1.1 Inlet Conditions

For laminar inlet profiles, often an analytical solution for fully developed flow is used. For example, the *setInletVelocity* tool [24] in OpenFOAM® can be used for this purpose. A transient inlet profile to model turbulent fluctuations can be realized by mapping back the flow in certain region to the inlet. The goal of such a mapping procedure is to find a simple way to describe the inlet conditions of a transient flow so that it looks as if it is turbulent [23].

Tabor and Ahmadi [23] compared different approaches for simulating such inlet conditions in the context of large eddy simulations. They distinguished between (i) synthesis methods and (ii) precursor simulations (e.g. internal mapping). Synthesis methods model the inlet condition of the flow, while precursor simulations are real simulations of the inlet condition running separate, before, or simultaneously to the actual simulation case. Based on this it was found that the best approach would be an internal mapping method: in this method the flow parameters in a source plane downstream of the inlet are sampled, corrected (e.g., $u_{mean} = 1$ is enforced), and finally mapped back to the inlet [25]. The method is simple, and has been successfully applied to square and circular cross section channels. According to Tabor and Ahmadi [23] there was no evidence for problems in the mapped section due to the usage of this method. Note, internal mapping is a simple technique which is already implemented in OpenFOAM® (see the "*pitzDaily"* tutorial case), and can therefore be used "out of the box" with no further modifications.

2.2 DEM Basics

The term DEM (short for Discrete Element Method) is used to describe a tool that is used to determine the translational and rotational motion of many particles in a dense granular system by solving Newton's equation of motion. Often, DEM is specifically used for a simulation tool that uses spring/dashpot/slider models to account for inter-particle contact forces, i.e., DEM is a so-called "soft-sphere" method. The forces appearing in the model are frequently determined by a highly appreciated, linear spring-dashpot force model [26]. Until now the focus in DEM research was mainly on spherical particles. Only in recent times the interest for simulating non-spherical particles increased. This is primarily because of the wide variety of different fields of application of the DEM, as well the observed different properties of granular matter consisting of spherical and non-spherical particles [27].

2.2.1 Non-spherical Particles

It is now well accepted that the behavior of particles in granular systems is strongly affected by their shape. Unfortunately, the knowledge gained from systems consisting of rigid, spherical particles cannot be directly extrapolated onto systems involving rigid non-spherical particles. Hence, the particle shape must be taken into account, which leads to a higher complexity of the simulation model. For example, for non-spherical particles the orientation is fundamental. Apart from the translational and rotational motion, contact possibilities must be described and the orientation needs to be updated frequently in the simulation. To still be comfortable with detecting particle-wall and particle-particle contacts for more complex non-spherical geometries, the method of spherosimplicies is recommended. Spherosimplicies are a tool to describe a complex geometry by a skeleton of multiple combined geometric elements, e.g., a set of lines [28]. In case spherosimplicies are adopted for the representation of a rigid fibre (assumed to be a spherocylindric object), only a line segment skeleton for each fibre needs to be tracked. Each line segment, represented by a point and a direction/shape vector, can be used to determine collisions with walls, and/or other particles [29]. For a particle-wall interaction, the contact point is detected by a line-triangle approach. In case of a particle-particle interaction, the contact point is detected analogous by a line-line approach [30]. The collision can then be numerically resolved by using the well-established linear spring-dashpot-slider modeling approach [26].

As an addition to the linear spring-dashpot model (which only predicts a force for overlapping particles), Lindström [31] describes the usage of a roughness layer around the fibre. This roughness layer model is meant to simulate a transition region between a non-interaction situation and an overlap situation. Specifically, two approaching fibres (only interacting with each other within their roughness layer) experience a repulsive force, which in some cases can even prevent an overlap. This roughness layer model might be interpreted as the effect of fibrils located on the fibres' surface [32].

DEM is not necessarily limited to rigid particles: flexible fibres, built as a chain of linked segments, have been investigated in the last two decades [33]–[35]. The linked segments making up a flexible fibre particle require the solution of a set of additional equations, thus requiring more computational power, or the investigation of smaller systems.

2.3 Coupling

The particles motion in a dry DEM simulation is influenced by contacts between the particles, and particles with the surrounding walls. In a two-phase (i.e., fluid-particle) CFD-DEM simulation the particles' motion is also influenced by the surrounding fluid acting on the particle. Lindström [31] describes the hydrodynamic forces originating from the presence of the fluid depending on the particle Reynolds number $Re_p=\rho UD/\mu$. For small Re_p the hydrodynamic forces are dominated by viscous effects; for large Reynolds numbers (i.e., $Re_p >> 1$) inertial effects are important. The equations used by Lindström [31] are built on the model developed by Cox [36]. The latter allows the conversion from a cylinder to a prolate spheroid, which has the same flow behavior in a shear flow. Note, in these hydrodynamic force equations, the force and the torque from the fluid acting on the particle are determined by using the velocity and angular velocity of the fluid and the particle at the same position. This position is the center of mass of the particle. This means that the shear rate gradient along the fibre is assumed to be small. Furthermore, Lindström [31] used a simplified simulation approach consisting only of a fluid-to-particle (i.e., one-way) coupling: the influence of the particles on the fluid was neglected. To reduce the inaccuracy of this simplification, a lubrication force model has been used. This model quantifies the force that originates from the squeezing motion of the liquid between two approaching fibres. The lubrication force model hence slows down two approaching fibres, facilitating fibre-fibre agglomeration, and hence flock formation.

3 Simulation Strategy

3.1 Fluid Flow near a T-Junction

3.1.1 T- Junction Geometry

The T-junction geometry used in all of our hydrodynamic fractionation simulations consists of a main channel and a side channel (see Fig. 2). The length of the inlet L_{inlet}^{+}, the side channel width SCW^{+}, and the angle α between the main channel and the side channel may vary. The height and the width of the main channel, i.e., $L^{+} = 1$, as well as the side channel length, i.e., $L_{side}^{+} = 6.0$, and the main channel outlet length, i.e., $L_{outlet}^{+} = 6.0$, are held constant in the present study.

The fluid flow direction in the main channel is in the positive x^{+}-direction, and in the side-channel in the negative z^{+}-direction. The fluid enters the T-junction at the inlet (in_C) located at the left side of the main channel, and exits the T-junction either through (i) the main channel outlet (out_C) on the right, or (ii) the side channel outlet (out_J) at the bottom. For the base case configuration ($L_{inlet}^{+} = 20$, $L_{outlet}^{+} = L_{side}^{+} = 6$, $SCW^{+} = 1$, $\alpha = 90°$), the origin of the global coordinate system is located at the intersection of the center lines of the main and the side channel. Note, decreasing the side channel width SCW^{+} leads to an increase of the inlet length L_{inlet}^{+} such that the outlet length L_{outlet}^{+} is kept constant.

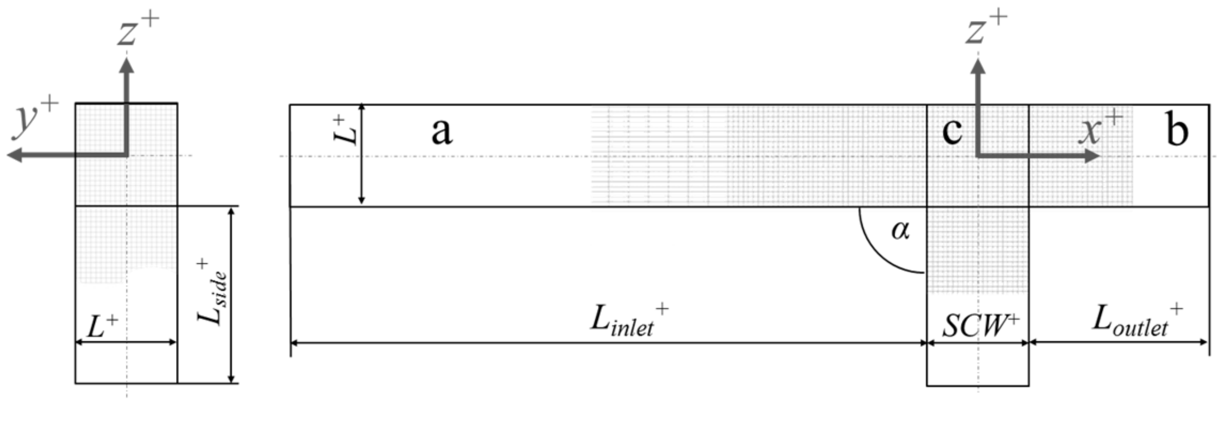


Fig. 2: T-junction geometry. Left panel: front view of the T-junction. Right panel: side view of the T-junction. Grey lines: qualitative representation of the mesh cells' size and shape.

3.1.2 Meshing

The first step for the investigation of fluid flow in a T-junction is the generation of a mesh, i.e., to split the simulation region in small regions in which flow variables are stored. As shown in Fig. 2, the geometry is split in three sections *a*, *b* and *c*. The side channel is generated by extruding section *c* in the negative z^+-direction. The inlet channel's base length is two length units. Longer inlets are generated by extruding section *a* in the negative x^+-direction.

First two meshes ($2.6 \cdot 10^6$ cells and $8 \cdot 10^6$ cells, $L_{inlet}^+ = 2$, $L_{outlet}^+ = L_{side}^+ = 6$, $SCW^+ = 1$, $\alpha = 90°$) are generated with the OpenFOAM® mesh generation tool *snappyHexMesh*. These meshes featured a refinement at the edges of the geometry to reduce the overall cell count. The meshes are used for simulations with $Re = 5{,}000$ and a (static) pressure difference of $\Delta p^+ = -0.30$ (the pressure difference between side channel outlet and main channel outlet is $\Delta p^+ = p_{out,J}^+ - p_{out,C}^+$). Unfortunately, when using such a mesh, an asymmetric (mean) flow profile was predicted (see Fig. 3).

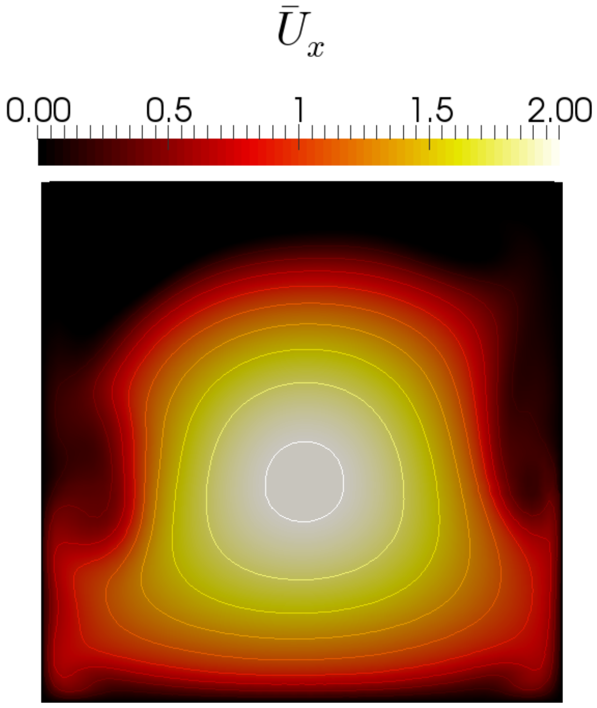


Fig. 3: Unsymmetrical fluid flow profile when using a refined mesh. Time-averaged velocity in main channel x^+-direction, shortly after the junction in the T-junction (Re = 5,000, $\Delta p^+ = -0.30$).

The calculations run 300 dimensionless time units to reach a quasi-steady state, then another 300 dimensionless time units to collect time-averages of mean and fluctuating flow properties. Both fluctuations and mean values are not fully symmetric in the y^+z^+-plane (in the main channel), or in the x^+y^+-plane in the side channel. This might be the result of the oblique polyhedron-cells generated by the *snappyHexMesh*-tool [37], or (less likely) too short averaging periods. This problem still appeared after mirroring the mesh and changing the decomposition of the geometry domain. Thus in the next step coarser meshes with no refinements at the edges were generated with the OpenFOAM® meshing tool *blockMesh*. This tool decomposes the T-junction geometry in hexahedral blocks [37]. Following such a mesh generation strategy, the symmetry problems vanished.

Regarding the main goal of this thesis, i.e., the investigation of the mechanism behind hydrodynamic fractionation, it was decided that for all further simulations meshes generated with *blockMesh*, with approximately $2.0 \cdot 10^5$ cells should be used. Also, the Reynolds number was kept constant at $Re = 500$, while the dimensionless pressure difference (Δp^+) was varied. The inaccuracy introduced due the use of these coarse meshes (the largest dimensionless wall distance [20] y^+-values are: $y_{min}{}^+ = \sim 0.7$, $y_{max}{}^+ = \sim 2.4$, $y_{ave}{}^+ = \sim 1.2$, see Appendix A) is expected to be low for the moderate Reynolds number studied. The low cell number decreases the computational time of each simulation, which simplifies an explorative investigation of multiple parameter settings.

Note, that in OpenFOAM® the Reynolds number *Re* was adjusted by changing the kinematic viscosity ($Re = \mathbf{u}\ \mathrm{H_{hyd}} / \nu$ with the hydraulic diameter: $H_{hyd} = 4A / U = L^+$ [14]). Thus $Re = 500$ required setting $\nu = 2 \cdot 10^{-3}$.

3.1.3 Numerical Schemes

In this thesis the fluid flow is solved using a direct numerical simulation (DNS) approach, and the finite volume method (FVM) is applied. In this method each cell of the generated mesh represents a single control volume, and all fluxes over the side faces of this control volume are explicitly determined. This leads to the main advantage of this method over other methods (e.g. FDM finite different method, FEM finite element method): its ability to fully satisfy mass, momentum and energy conservation [38].

OpenFOAM® approximates the Navier-Stokes-Equations (see Appendix A) by applying discretization schemes, and the subsequent solution of the set of algebraic equations. Those schemes can be divided in time and spatial discretization. In Table 1 the numerical schemes used for all CFD simulations in this thesis are summarized. Furthermore, the OpenFOAM® standard solver *pimpleFOAM* was applied, which allows a flexible adjustment of the velocity-pressure segregated solution procedure employed by OpenFOAM®.

Table 1: Numerical schemes used in OpenFOAM®.

Time Discretisation

keyword	math. terms	schemes	
ddtSchemes	First/Second order time derivative ∂/∂t	default	backward

Spatial Discretisation

keyword	math. terms	schemes	
gradSchemes	Gradient ∇, e.g. ∇p	default	Gauss linear
		grad(p)	Gauss linear
		grad(U)	Gauss linear
divSchemes	Divergence∇·, convection term, e.g., ∇·(ρ**uu**)	default	none
		div(phi,U)	Gauss limitedLinearV 1
		div(phi,nuTilda)	Gauss limitedLinear 1
		div((nuEFF*dev(T(grad(U)))))	Gauss linear
laplacianSchemes	Laplacian ∇², laplacian terms, e.g., ∇·(ν∇**u**)	default	none
		laplacian(nuEff,U)	Gauss linear corrected
		laplacian(rAUf,p)	Gauss linear corrected
interpolationSchemes	Point-to-point interpolations of values, interpolation from cell centers to face centers	default	linear
		interpolate(U)	linear
snGradSchemes	component of gradient normal to a cell face	default	corrected
fluxRequired	fields which require the generation of a flux	default	no
		p	

For detailed information regarding the mathematics behind the chosen discretization schemes see the OpenFOAM® user guide [37].

3.1.4 Boundary Conditions

The boundaries of the simulation domain are the main channel inlet (in_C), the main channel outlet (out_C), the side channel outlet (out_J) and the wall building the T-junction geometry. The conditions used for the pressure and velocity can be found in Table 2 and Table 3.

Table 2: T-junction: fluid pressure boundary conditions.

patch	type	inletValue		Value	
inC	zeroGradient	-	-	-	-
outC	fixedValue	-	-	uniform	0
outJ	fixedValue	-	-	uniform	[-0.3, -0.2, -0.1, 0.0, 0.1, 0.2, 0.25, 0.30, 0.35]
wall	zeroGradient	-	-	-	-

Table 3: T-junction: fluid velocity boundary conditions.

patch	type	inletValue		Value	
inC	see Chapter 3.1.4.1 and 3.1.4.2				
outC	inletOutlet	uniform	(0 0 0)	uniform	(0 0 0)
outJ	inletOutlet	uniform	(0 0 0)	uniform	(0 0 0)
wall	fixedValue	-	-	uniform	(0 0 0)

Note, for all cases gravity is neglected, meaning that only the dynamic pressure in the channels is predicted, and that sedimentation effects are absent in our simulations.

3.1.4.1 Laminar Inlet

In most simulations a laminar inlet profile is applied at *inC*. The profile is generated with the OpenFOAM® tool *setInletVelocity* [24] (see Appendix A). Fig. 4 shows the velocity profile that has been imposed at the inlet of the T-junction.

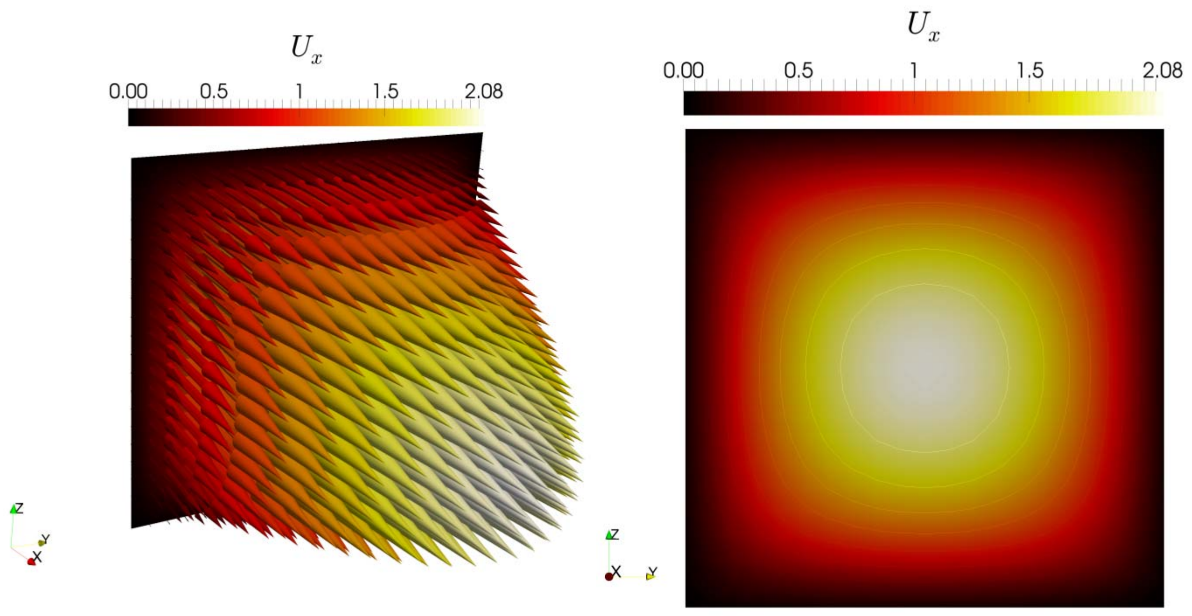


Fig. 4: Laminar inlet profile. Left panel: isometric view. Right panel: y^+z^+-plane of main channel inlet.

3.1.4.2 Mapped Inlet

Also a mapping technique for representing multiple T-junctions was investigated. Such a mapping strategy allows us to picture the effect of unsteady and inhomogeneous flow generated at a T-junction on the separation efficiency. Specifically, a sample field downstream is corrected (e.g., the mean inlet velocity is enforced to be 1), and mapped back to the inlet of the T-junction. Fig. 5 shows the geometry of the base case T-junction with the mapped region. The sample field for the mapped inlet condition is represented by the grey-shaded plane. The settings for the mapped boundary condition [25] (see Appendix A) are based on the *pitzDaily* tutorial case of OpenFOAM®.

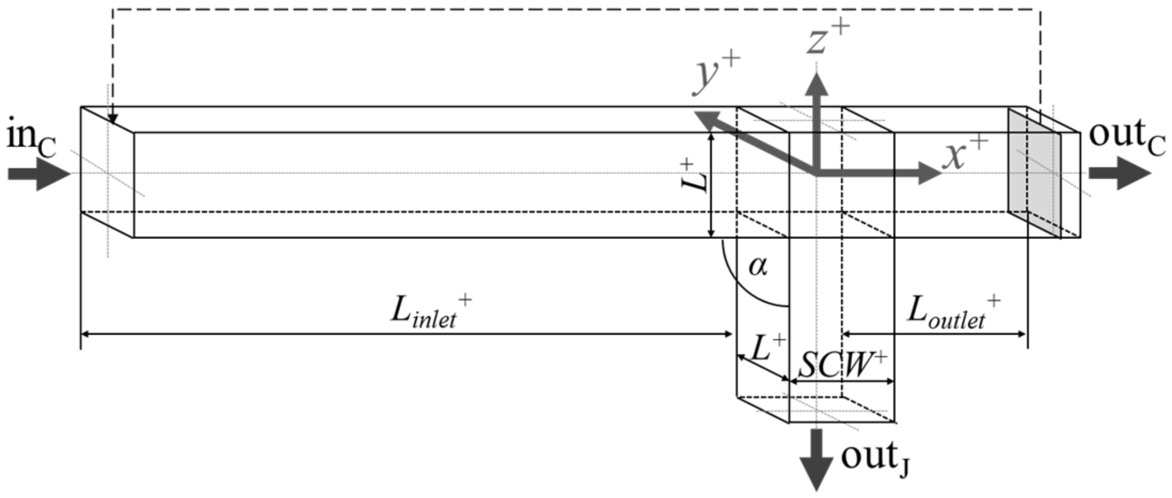


Fig. 5: Internal mapping to generate the inlet flow profile. The grey-shaded plane represents the cross section which is mapped back to the inlet.

At a low Reynolds number (i.e., *Re* = 500) the fluid flow in the T-junction with a mapped inlet condition and a laminar inlet condition are very similar (see Appendix A). Thus, for all further simulation no internal mapping was applied.

Note, due to the low Reynolds number and the laminar inlet profile, the mesh cell number along the inlet can be reduced, since the flow will be developed and laminar. This again saves computational time.

In addition to the T-junction geometry, a straight channel geometry was generated and used for an explorative investigation of the fibre behavior in laminar flow. For the corresponding details the interested reader is referred to Appendix A.

3.2 Particle Motion

To be able to simulate particle separation in a hydrodynamic fractionation device, the particles, i.e., fibres, need to be modeled. Therefore, the geometry of a single fibre and the forces acting on it need to be defined. In general, the fibres' motion is influenced by fibre-wall, and/or fibre-fibre contact forces, as well as hydrodynamic forces. The latter results from the surrounding fluid, and is determined by sampling fluid data at each fibre's center of mass position.

3.2.1 Fibre Model

In order to determine fibre-fibre or fibre-wall contacts, the fibres are assumed to be spherocylinders. Note, that for all other calculations, e.g. force models, the fibres are assumed to be prolate spheroids (see Fig. 6). This is done in order to save computation time, since the calculation of interaction of spherocylinders (or a spherocylinder with a wall) is faster compared to that of spheroids [28]. Thus, the geometric assumptions about the exact fibre shape are strictly speaking inconsistent. However, it is speculated that this inconsistency will have little effect on the overall behavior of the system, since we are mainly interested in particles with high aspect ratios.

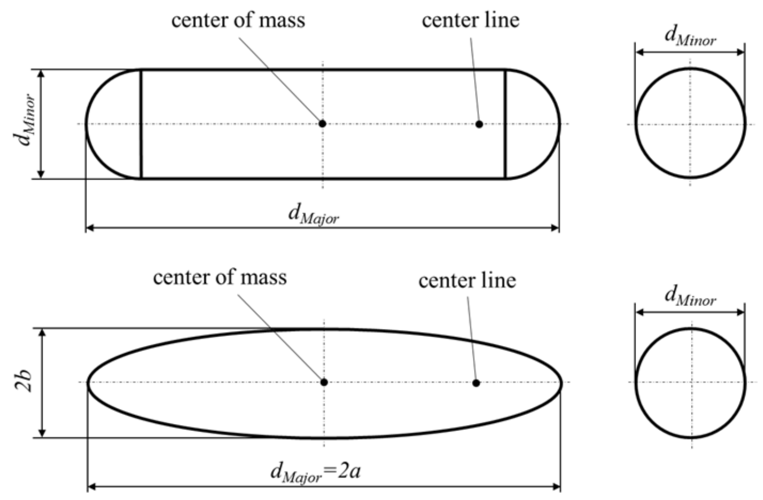


Fig. 6: Fibre geometry assumptions. Top panel: spherocylinder. Bottom panel: prolate spheroid.

3.2.2 Fibre-Fibre and Fibre-Wall Interactions

The implementation of the fibre-fibre and fibre-wall contact model is based on the algorithms present in Schneider and Eberly [30]. The spherocylinders are represented via a method based on spherosimplicies [28], i.e., as a line segment. Each line segment is defined by a point and a direction, while the surrounding walls can be seen as an ensemble of triangles [29]. The used fibre-fibre algorithm determines the nearest distance between two lines (i.e., the centerlines of the spheroids), as well as the points on these lines that have the nearest distance to each other. Similarly, the fibre-wall algorithm determines the distance between a line and a triangle. Fibre-fibre and fibre-wall forces are modeled with the Discrete Element Method (DEM). Our model accounts for an inelastic interactions in the normal direction, as well as a dashpot and frictional element in the tangential direction [39]. Furthermore, the fibres are assumed to be surrounded by a roughness layer, which represents the fibrils of a fibre [32] (see Fig. 7). The effect of fibrils is modeled using a normal linear spring force model. Based on this roughness, and the viscosity of the surrounding fluid, the effect of unresolved fluid flow between two fibres (or a fibre and a wall) can also be modeled. This so-called lubrication force model establishes a force against a possible relative motion of the fibres, i.e., it dampens fibre relative motion [31]. For details regarding these models the reader is referred to Appendix B.

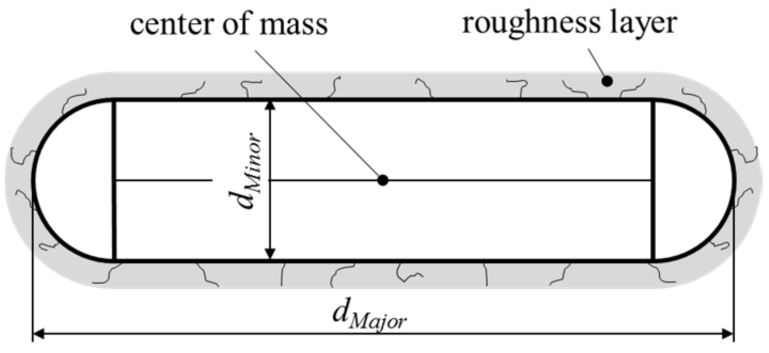


Fig. 7: Fibre with roughness layer to represent fibrils.

The correct implementation of the fibre-fibre interaction models is verified by comparing simulation results with an analytical solution of a single fibre-fibre collision (see Appendix C for details). The fibre-wall interaction has been verified by Redlinger-Pohn [29] using a similar strategy, and is not discussed here in detail.

3.2.3 Fibre-Fluid Interactions

In our model the fibres are influenced by hydrodynamic forces and torques acting on the center of mass, while the influence from the fibres on the fluid is neglected. Depending on the particle Reynolds number Re_p, the hydrodynamic forces are dominated by viscous (small Re_p) or inertial effects (large Re_p) [31]. In this work no inertial effects are taken into account since Re_p is small. Furthermore, in the current work no lubrication force model (see chapter 3.2.2) is used to simulate the resistance two approaching fibres (or a wall-approaching fibre) experience. This was done due to the lack of information regarding the fibrils presence on the surface of the fibres.

3.2.4 Coordinate System

3.2.4.1 Global and Fibre Coordinate System

In all DEM and CFD-DEM simulations three coordinate systems have to be distinguished (see Fig. 8):

- The global coordinate system used by LIGGGHTS® **B** ($\mathbf{e}_x$, $\mathbf{e}_y$, $\mathbf{e}_z$)
- The cross section section's coordinate system $\mathbf{B}_i''$ ($\mathbf{e}_{x,i}''$, $\mathbf{e}_{y,i}''$, $\mathbf{e}_{z,i}''$)
- The fibres coordinate system $\mathbf{B}_{j,i}'$ ($\mathbf{e}_{x,j,i}'$, $\mathbf{e}_{y,j,i}'$, $\mathbf{e}_{z,j,i}'$)

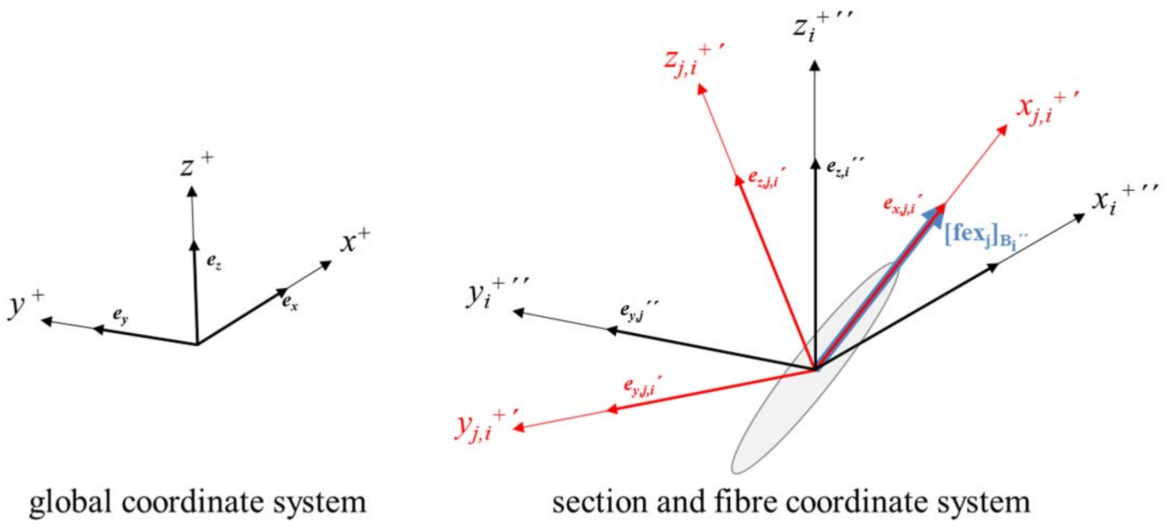


Fig. 8: Coordinate systems used in this study. Left panel: global coordinate system. Right panel: local cross sectional coordinate system, as well as fibre coordinate system.

Transformations between coordinate systems are straight forward using matrix algebra (see Appendix C for details).

3.2.4.2 Section-Based Coordinate System

To analyze the orientation of the fibres, the cross section of the channel is divided into four sub-sections (see Fig. 9). This is mainly done to compare fibre orientation statistics (obtained from the channel flow simulations) to literature data [40] on fibre flow between two parallel walls. In our work the global coordinate system **B** ($\mathbf{e}_x$, $\mathbf{e}_y$, $\mathbf{e}_z$) is located at the center of the simulation box. The other sections' coordinate systems $\mathbf{B}_i''$ ($\mathbf{e}_{x,i}''$, $\mathbf{e}_{y,i}''$, $\mathbf{e}_{z,i}''$) are located on each side wall, and have their origin at the center of the cross section (see Appendix C).

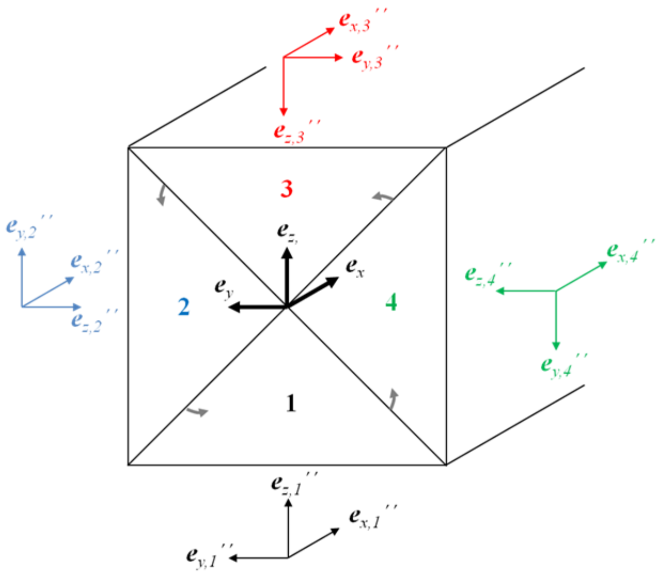


Fig. 9: Simulation box, divided into 4 equally sized triangular sections 1, 2, 3 and 4. Each section *i* has its own section-based coordinate system B$_i''$ (e$_{x,i}''$, e$_{y,i}''$, e$_{z,i}''$). Middle: Global coordinate system B (e$_x$, e$_y$, e$_z$).

3.2.4.3 Fibre Orientation in a Spherical Coordinate System

The fibre orientation vector in each section *i* (**[fex$_j$]$_{Bi''}$** referring to each sections' base point **B$_i$**$''$) is converted from Cartesian coordinates to spherical coordinates (see Fig. 10, detailed information on the conversion operation is summarized in Appendix C). This is done in analogy to the analysis of fibre orientation in suspension flow between two parallel plates [33].

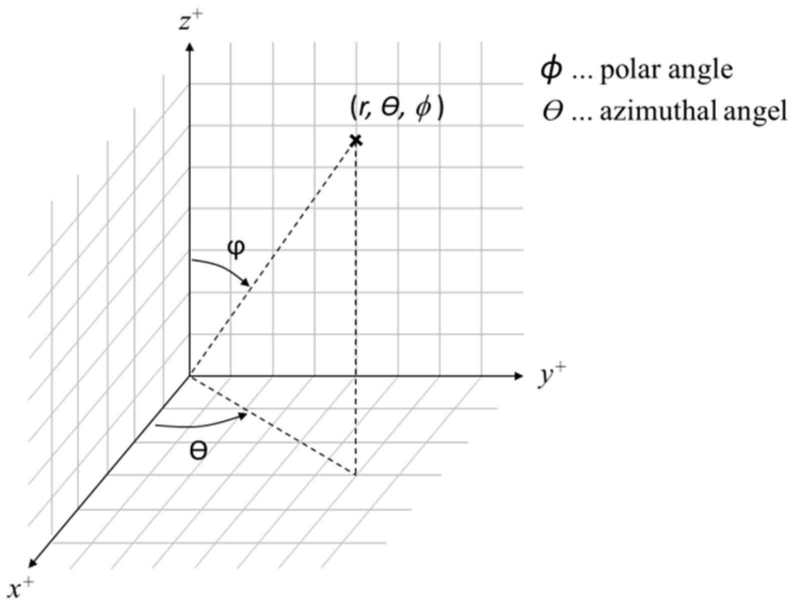


Fig. 10: Spherical coordinate system [41].

To illustrate the fibre orientation analysis (i.e., [Θ$_j$]$_{Bi''}$ and [ϕ_j]$_{Bi''}$), three possible extreme orientations of a fibre are considered and displayed in Fig. 11.

- the fibre is oriented in the global x^+ – direction: streamwise oriented
- the fibre is oriented in the global y^+ – direction: parallel to wall
- the fibre is oriented in the global z^+ – direction: normal to wall

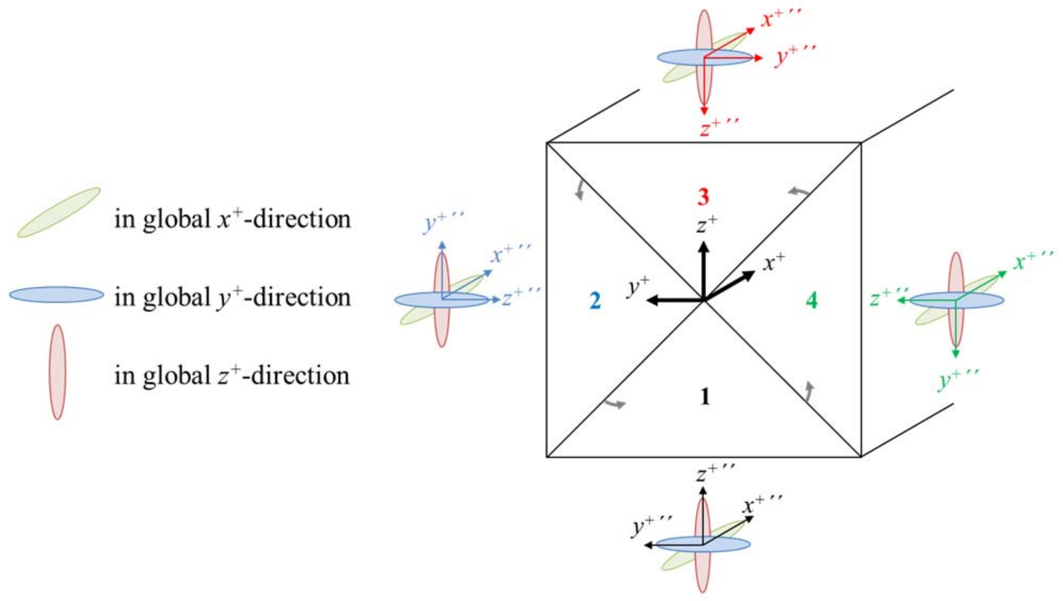
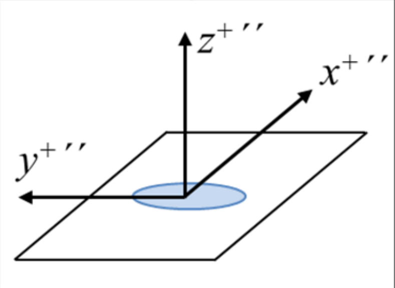
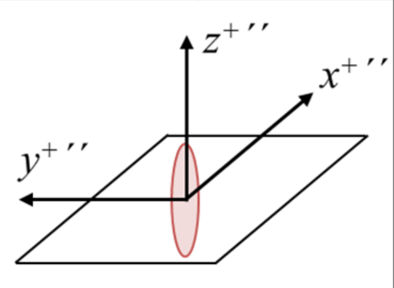
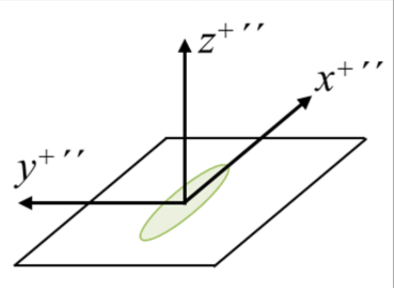


Fig. 11: Three possible extreme fibre orientations in each sub-section of the cross section.

Depending on the sub-section the fibre resides in, a different azimuthal Θ and polar angle ϕ is obtained, since the fibre has a different position relative to the wall. In Table 4 the three possible fibre orientations: parallel to the wall (||, spanwise), normal to the wall (∟**),** or directed into streamwise direction (*sw*) are shown.

Table 4: Analysis of the three possible extreme fibre orientations in terms of spherical coordinates.

|| ... parallel to the wall		**∟ ... normal to the wall**		***sw* ... streamwise direction**	
$\Theta = \pm 90°$	$\phi = 90°$	Θ = all	$\phi = 0° / 180°$	$\Theta = 0° / 180°$	$\phi = 90°$
					

3.2.5 Benchmark Simulation: Fibre Flock Formation

In the pulp and paper industry flocculation of fibres occurs already at comparably low consistencies [35], the key additional factor being the fibres' aspect ratio. Thus the simulation of a fibre flock generation with interacting fibres seems to be a logical first step to test the fibre-fibre interaction model.

Previous work ([34], [35], [42], [43]) focused on flexible fibres for the generation of fibre flocks. The flexibility of fibres needs additional equations to be solved for each fibre, which results inevitably in the (possibly) limiting requirement of high computational power. The usage of stiff fibres takes fewer equations, and thus no time-expensive simulations, which is advantageous especially for the investigation of the fibre flock generation technique itself. In this study, the formation of a fibre flock consisting of stiff fibres was investigated.

3.2.5.1 Representation of the Fibre Length Distribution

In all DEM and CFD-DEM simulations five fibre classes are used, whereas the fibre classes differed only in their major length (i.e., in their aspect ratio). The minor length is constant for all fibres. To compare the fibre dimensions used in the simulation with reality, a reference system is used: 1 dimensionless length unit = $5 \cdot 10^{-3}$ [m]. Furthermore, two fibre distributions need to be distinguished, the real fibre distribution $\Delta Q_{0,real}$ measured by König [44] and a uniformly distributed distribution $\Delta Q_{0,uniform}$.

Table 5: Fibre types: dimensions and distributions.

			in simulation [-]		in ref. system [m]	
AR	$\Delta Q_{0,real}$	$\Delta Q_{0,uniform}$	d_{Major}	d_{Minor}	d_{Major}	d_{Minor}
2	0.95	0.20	0.04	0.02	$2.0 \cdot 10^{-4}$	10^{-4}
5	0.01	0.20	0.10	0.02	$5.0 \cdot 10^{-4}$	10^{-4}
10	0.03	0.20	0.20	0.02	$1.0 \cdot 10^{-3}$	10^{-4}
15	0.01	0.20	0.30	0.02	$1.5 \cdot 10^{-3}$	10^{-4}
20	0.01	0.20	0.40	0.02	$2.0 \cdot 10^{-3}$	10^{-4}

3.2.5.2 Tri- and Uni-axial Deformation

A flock of fibres has been prepared by a virtual compaction of a three-dimensional, fully periodic simulation box that is filled with randomly positioned and oriented fibres. While the box shrinks, the fibres get closer to each other, interact (due to the contact force model detailed in Chapter 3.2.2) and move relative to each other.

There are multiple options on how exactly the original box shrinks. Two possibilities were considered, namely (i) an uni-axial, and (ii) a tri-axial deformation method, both having a constant (negative) deformation rate ε_{deform}. While the uni-axial model deforms the box only in one spatial direction (i.e., the x^+-direction in our setup), the tri-axial deformation model applies a deformation to all three spatial directions (see Fig. 12).

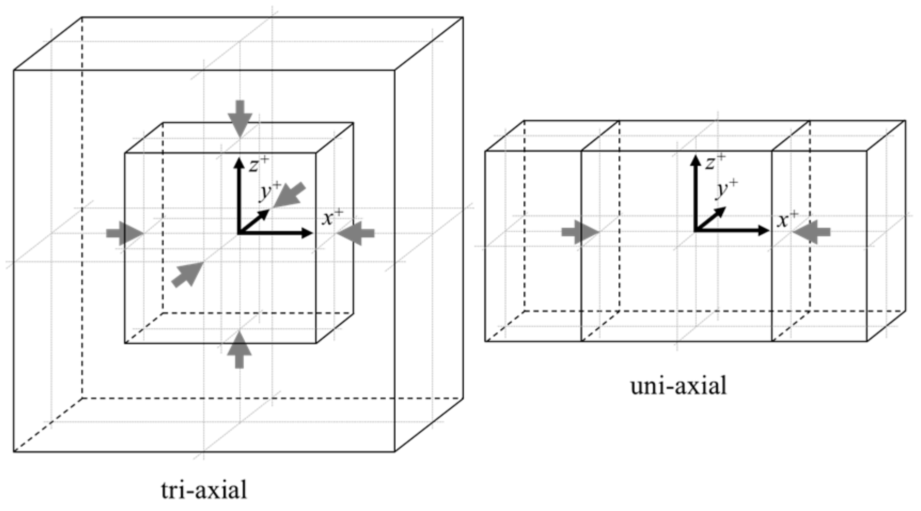


Fig. 12: Illustration of the compaction procedure to produce a fibre flock. Left panel: tri-axial deformation. Right panel: uni-axial deformation in x^+-direction.

To guarantee that all fibres fit in a certain region (e.g., in a channel), the original (pre-deformation) box is deformed to a reduced final (reduced post-deformation) box. The reduced deformed box dimensions are exactly the final (post-deformation) box dimensions minus the fibre length of the longest fibre in the system (i.e., in our case $d_{Major,AR=20}$, see the illustration in Fig. 13).

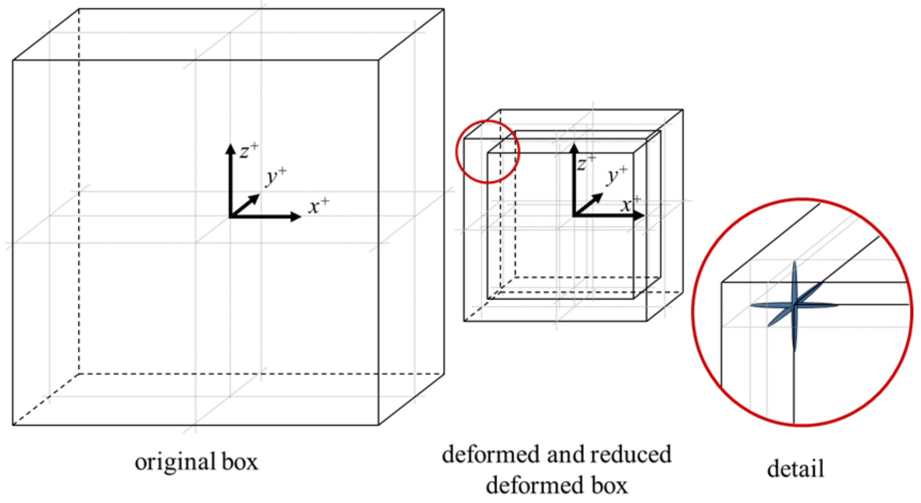


Fig. 13: Dimensions of the box used for the compaction procedure. Left: original box. Middle: final (post-deformation) box and reduced final (reduced post-deformation) box. Right: detail: fibre in the corner of the reduced final box.

Note, after the deformation process a stabilization process is started. Therefore the simulation continues with no deformation but a repeating resetting of the fibres' velocities. After a certain number of time steps a sufficiently low total internal energy of the system (e.g., 10^{-20})

is reached. This indicates that all fibres have positioned in an (approximately) stressfree arrangement in the flow.

3.2.5.3 Fibre Flock Generation Setup

In this case following parameter settings are used. The fibres are uniformly distributed (see Table 5).

Table 6: Fibre flock generation setup for tri- and uni-axial deformation.

DEM Parameter		**Fibre/Fluid Data**	
roughFact	$1.0\cdot10^{-2}$	rho_{Fib}	1.27
roughStiffFact	$1.0\cdot10^{-3}$	phi_{Fib}	0.01
frictionCoef	0.0	rho_{Fluid}	1.00
liquidDynViscosity	0.0	**Tri-axial deformation**	
dt_{DEM}	$1.0\cdot10^{-6}$	direction	initial box
koverlap	$5.0\cdot10^{-3}$	x^+	50
compactEveryNthDEMStep	$5.0\cdot10^{1}$	y^+	50
$stiffness_{Normal}$	$2.0\cdot10^{4}$	z^+	50
$damping_{Normal}$	$1.0\cdot10^{-2}$	**Uni-axial deformation**	
$stiffness_{Tang}$	0.0	direction	initial box
$damping_{Tang}$	0.0	x^+	1000
$d_{MinorRealFact}$	1.0	y^+	4.6
$E_{tot,min}$	$1.0\cdot10^{-10}$	z^+	4.6

Table 6 summarizes the fibre properties of the fibre flock generated with these setting and the required deformation rates for the tri- and uni-axial deformation process.

Table 7: Fibre flock data for tri- and uni-axial deformation. "real"-values represent properties of the final fibre flock.

Fibre Data		**Box Dimensions**	
phiFibreal	$1.00\cdot10^{-2}$	direction	red. Final box
massFracReal	$1.27\cdot10^{-2}$	x^+	4.6
consitency	$1.27\cdot10^{-2}$	y^+	4.6
nFibres	22340	z^+	4.6
Tri-axial deformation		***AR***	$\boldsymbol{n_{Fib}}$
tRateValue	-5.0006251	2	4468
RealDEMsteps	477150	5	4468
Uni-axial deformation (only in x^+)		10	4468
tRateValue	-5.0006251	15	4468
RealDEMsteps	1,076,250	20	4468

Note, the spring stiffness and the damping coefficient are chosen similar to the fibre-fibre test case: the minimum number of time steps resolving a fibre-fibre contact is >50.

3.2.5.4 Analysis of the Fibres' Orientation in a Fibre Flock

Two fibre flocks generated with both deformation methods (tri- and uni-axial deformation) and a detail view of both fibres can be seen in Fig. 14. A difference in the fibres' orientation between those two methods can already be visually detected.

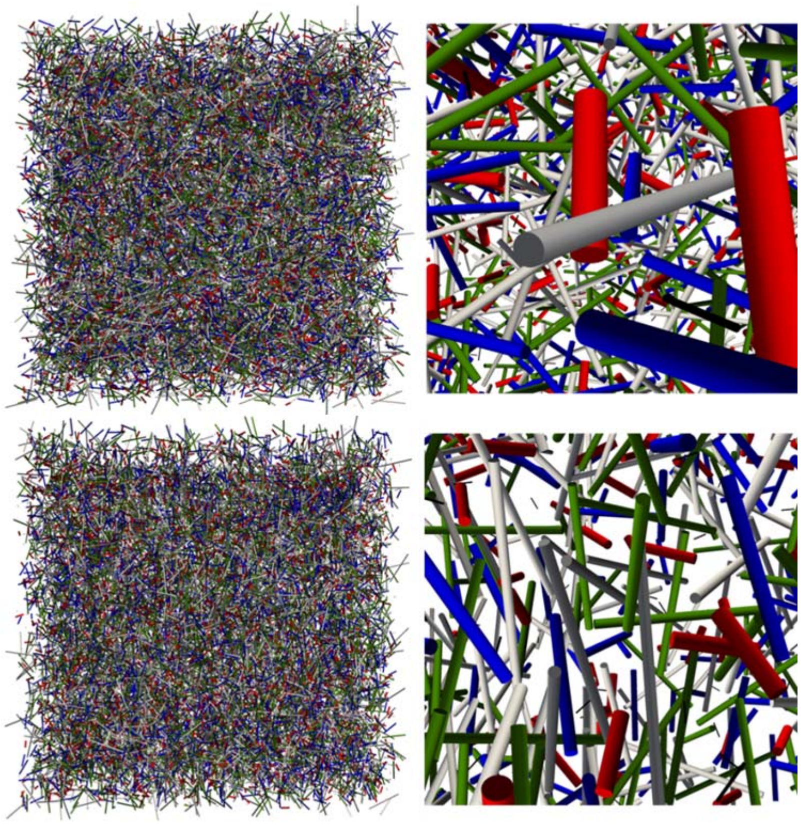


Fig. 14: Snapshots of the fibre flocks. Top: fibre flock generation with tri-axial deformation method. Bottom: fibre flock generation with uni-axial deformation method. Left: total fibre flock. Right: fibre flock detail.

For a more precise fibre orientation analysis, the fibre orientations at the beginning and at the end of the fibre flock generation process are compared (see Fig. 15 to Fig. 17). It can be seen that for the tri-axial deformation there is no orientation preference. Thus, an approximately uniformly distribution (compare with the data summarized in Appendix C) of the fibre orientation is kept during the domain deformation process. For the uni-axial deformation a preferred fibre orientation develops: the fibres in the uni-axial deformed box are more likely to be parallel to the walls, which is especially true for long fibres. Furthermore, it can be observed that the initial box size is crucial to the pronounced development of a preferred fibre orientation: the longer the initial x^+-dimension for the uni-axial method, the more dominant the preferred fibre orientation is. This is because the traveling length and the time for the orientation process are longer.

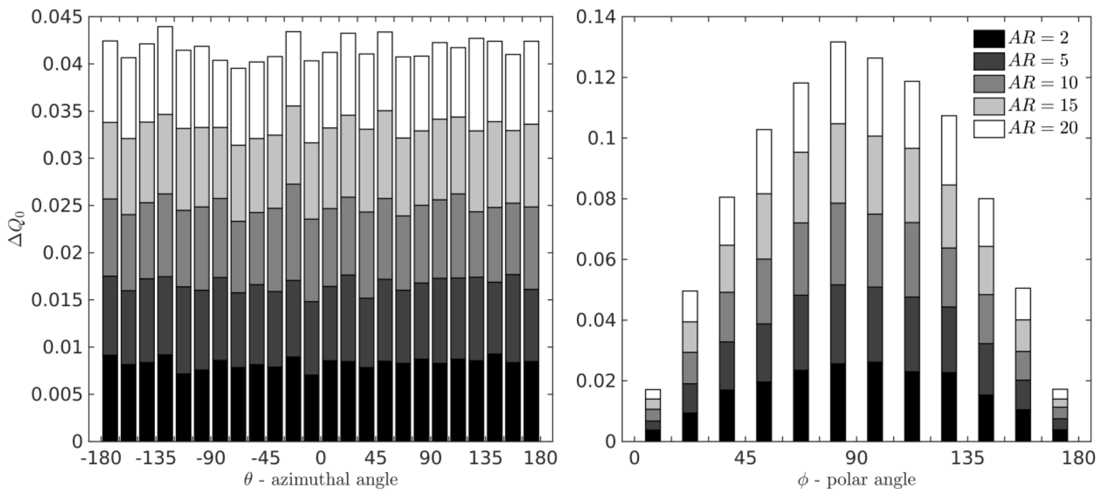


Fig. 15: Initial orientation distribution for the tri- and uni-axial deformation process.

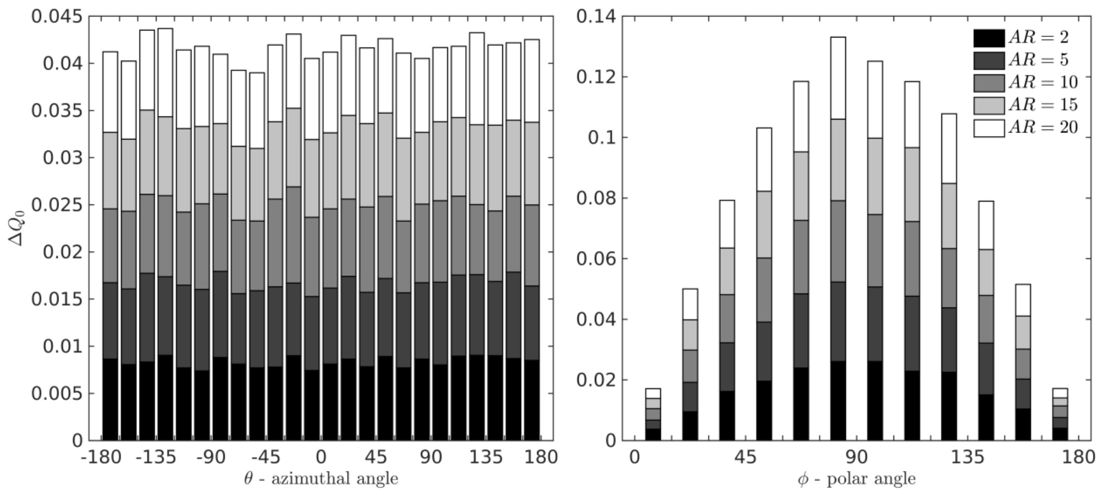


Fig. 16: Tri-axial deformation: fibre orientation at the end of the simulation, φ_{Fib} =1.0%.

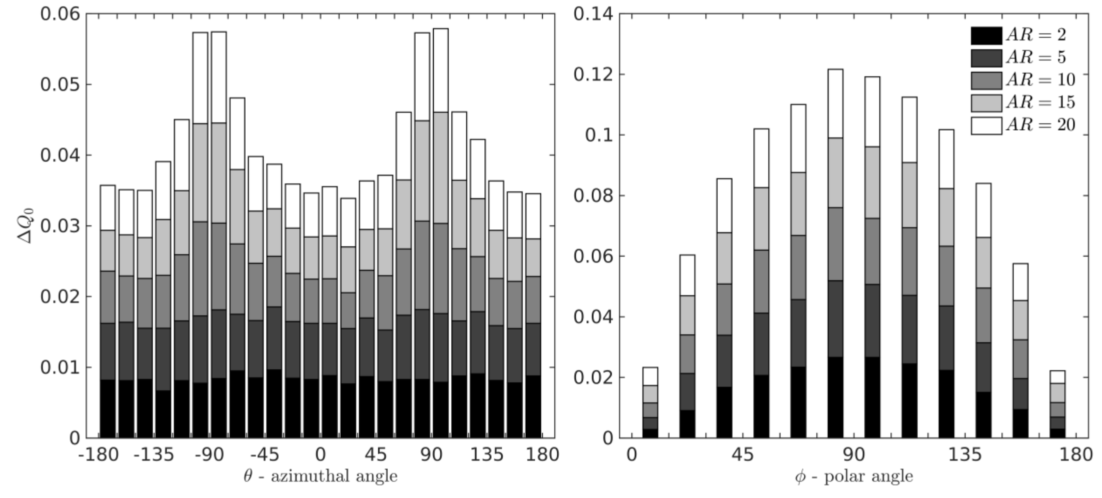


Fig. 17: Uni-axial deformation: fibre orientation at the end of the simulation, φ_{Fib} =1.0%.

Note, all classes have the same width of 15 degrees.

4 Results for Fluid Flow

To evaluate the behavior of the fluid in the T-junction at different parameter settings, multiple CFD-simulations are executed. For details on each simulation case see Appendix A. All simulations are divided into two parts: first, the simulation is executed until a quasi-steady state flow profile is reached (see Appendix A). Second, the simulations are continued to determine time-averaged pressure and velocity distributions. Furthermore, by spatially averaging the time-averaged velocity and pressure over the inlet and outlet boundary planes, it is possible to calculate the mean volumetric flow fraction Φ^+ leaving the T-junction via the side channel and the pressure loss in the main channel over a single T-junction.

4.1 Volumetric Flow Fraction exiting through the Side Channel

Depending on the geometry (L_{inlet}^+, SCW^+, α) and the pressure difference Δp^+ between side channel outlet and main channel outlet, the volumetric flow fraction Φ^+ leaving the T-junction through the side channel varies. It is assumed that the volumetric flow fraction Φ^+ is directly connected with the separation efficiency of the hydrodynamic fractionation. This is since a higher volumetric flow fraction results in a larger amount of fluid drawn into the side channel, and consequently also a larger fraction of particles is removed from the main channel.

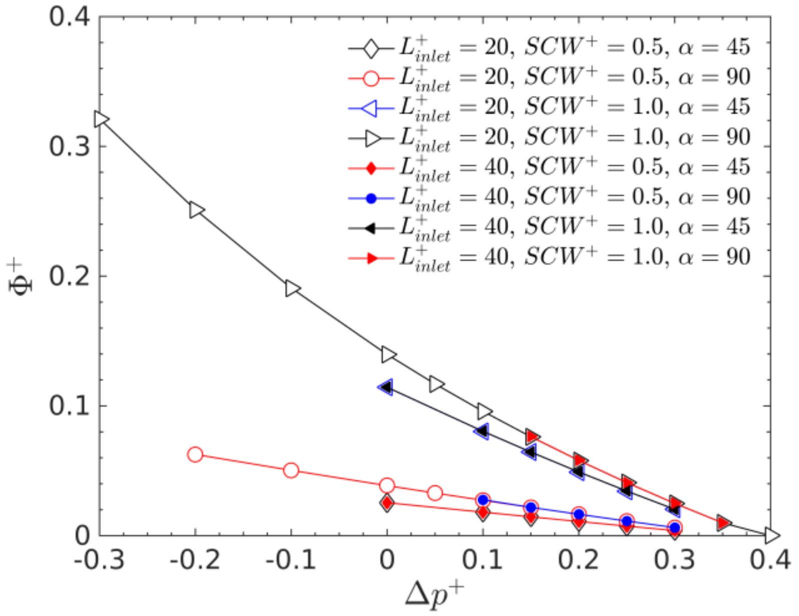


Fig. 18: Volumetric flow fraction leaving the T-junction via the side channel. *Re* = 500, $L_{outlet}^+ = L_{side}^+ = 6$.

Fig. 18 highlights the effect of pressure, angle, and inlet length on the flow rates. The main impact on the volume fraction Φ^+ is given by the pressure difference and the side channel width SCW^+. Clearly, a smaller width results in a smaller cross section of the side channel, and consequently Φ^+ decreases with decreasing SCW^+. The angle of the side channel is also influencing the volumetric flow fraction, but this effect is smaller than that of SCW^+. No influence (i.e., ~1‰) of the inlet length of the channel L_{inlet}^+ on Φ^+ is found, as it should be. In summary, predicting Φ^+ based on the geometrical and operating parameters of the T-junction seems non-trivial, asking for a more in-depth analysis.

4.2 Pressure Distribution near a T-Junction

To gain a more profound understanding of the flow behavior near the T-junction, the pressure distribution for three cases is shown in Fig. 19 and Fig. 20.

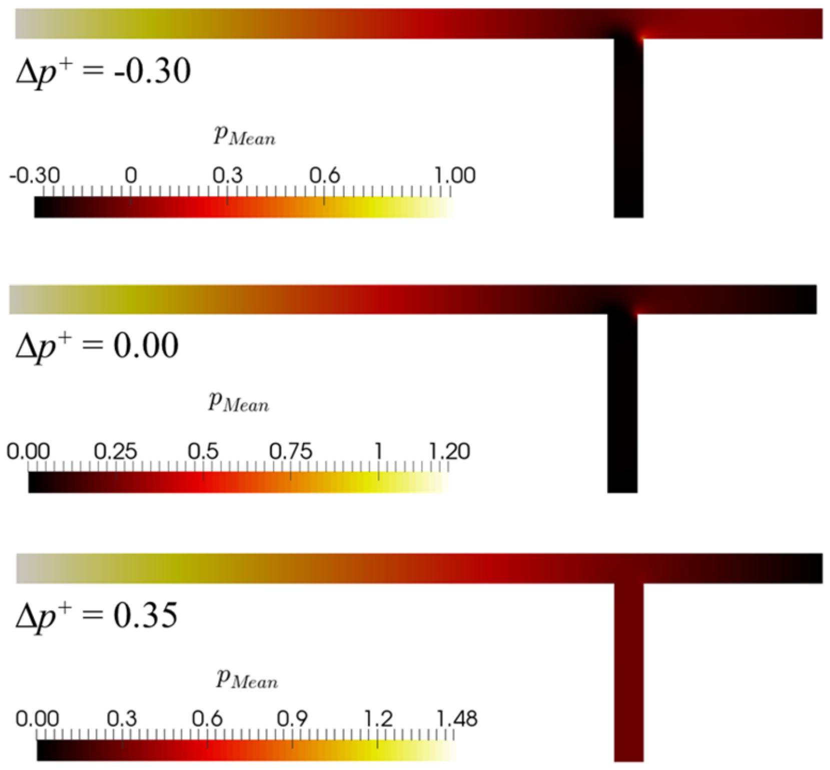


Fig. 19: Pressure distribution at a T-junction (L_{inlet}^+=20, L_{oulet}^+=L_{side}^+= 6, SCW^+=1, α=90°). Top penal: Δp^+=-0.30. Middle panel: Δp^+=0.00. Bottom panel: Δp^+=0.35.

The static outlet pressure (divided by the fluid density) is set via the boundary conditions in OpenFOAM®. In all cases the static outlet pressure at the main channel outlet is set to zero. Thus the pressure difference Δp^+ actually equals the static pressure at the side channel outlet.

In Fig. 19 three cases are compared. In all cases the highest total mean pressure is located at the inlet, as it should be to realize fluid flow through the T-junction. The top panel contains an under-pressure at the side channel, which drags fluid to the side channel. Consequently, the total mean pressure in the outlet part of the main channel is higher than in the side channel. In the middle panel (of Fig. 19) the static pressure at both outlets is identical. The bottom panel (of Fig. 19) shows an over-pressure configuration, i.e., the total mean pressure in the outlet part of the main channel is lower than in the side channel. Note, the total pressure consist of the static and the dynamic pressure [14], thus a direct comparison of the total pressure to predict the volumetric flow fraction is misleading. The velocity of the fluid in the channels, i.e., the dynamic pressure, need to be considered as has been done in the analysis detailed below.

Fig. 20 shows the pressure distribution over the side channel. As expected, the mean pressure decreases in the $-z^+$- direction, due to the pressure loss, which is according to Hagen–Poiseuille's equation [14] proportional to the length of the channel and the fluid velocity.

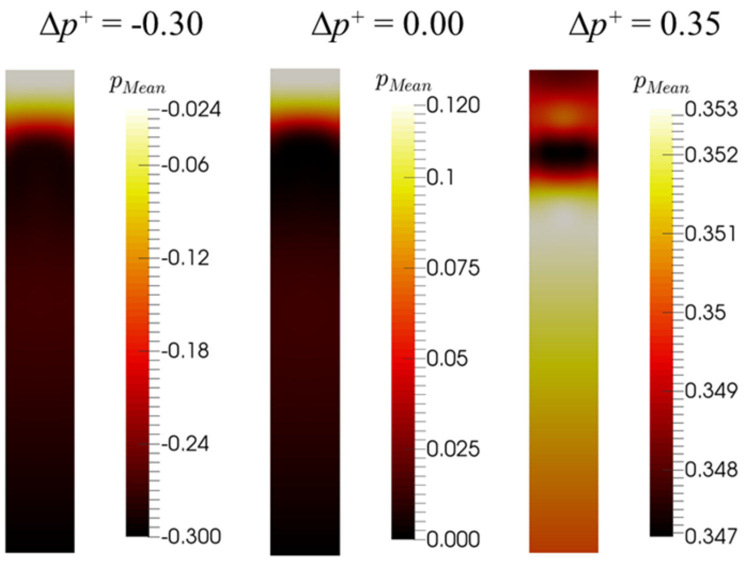


Fig. 20: Pressure distribution in the side channel of a T-Junction (L_{inlet}^+=20, L_{oulet}^+=L_{side}^+= 6, SCW^+=1, α=90°). Left penal: Δp^+=-0.30. Center panel: Δp^+=0.00. Right panel: Δp^+=0.35.

4.2.1.1 Total Pressure Loss in the Main Channel

For economic reasons the total pressure loss, i.e., the energy dissipation rate, of a hydrodynamic fractionation device is important. The pressure loss over a single T-junction (in the main flow direction) is determined for multiple geometries and pressure differences, and summarized in Fig. 21.

Qualitatively it can be noted that the main influence on the overall pressure loss seems to be the pressure difference Δp^+ and the inlet length L_{inlet}^+, followed by the side channel width SCW^+ and the angle α between the two channels. For details on the determination of the pressure loss, see Appendix A. Considering the scaling implied by the Hagen-Poiseuille pressure loss, i.e., $\Delta p \propto \bar{u}\, L\, \mu\, \zeta \,/\, H^2$ it is only logical that a longer inlet length and a higher fluid velocity results in a higher pressure loss. In summary, the higher Δp^+, the higher the flow rate through the main channel resulting in a higher pressure loss in the main channel. Note, that the pressure drop in the side channel has not been taken into account in the current analysis, which might shift the picture for the overall energy dissipation rate in T-junction flow.

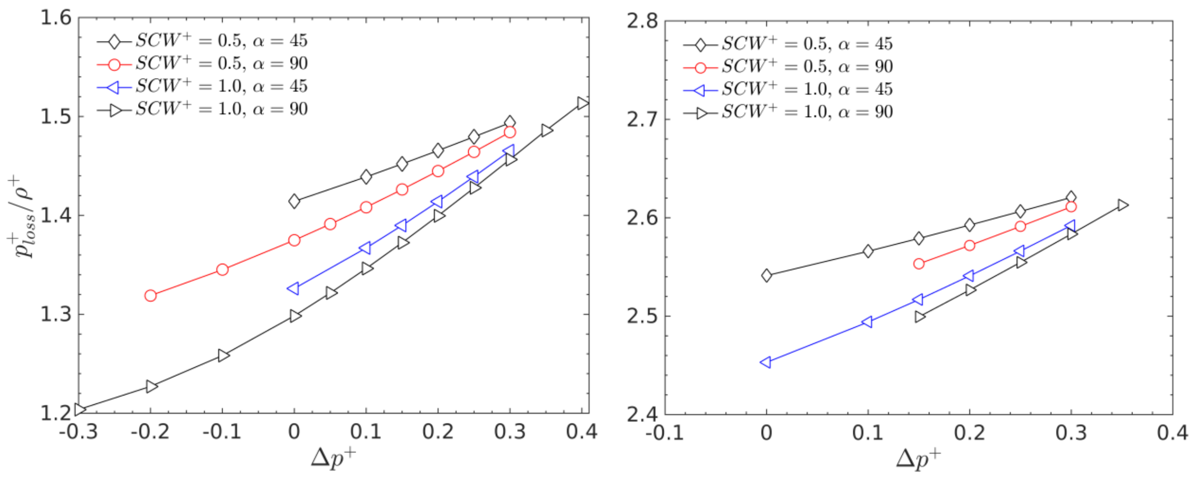


Fig. 21: Overall pressure loss over a single T-junction (Reynolds number *Re* = 500, outlet length L_{outlet}^+ = 6, inlet length L_{inlet}^+ and side channel width SCW^+ vary in these simulations; left panel: T-junction geometry with an inlet length of L_{inlet}^+ = 20 (SCW^+=1) and L_{inlet}^+= 20.5 (SCW^+= 0.5); right panel: T-junction geometry with an inlet length of L_{inlet}^+= 40 (SCW^+=1) and L_{inlet}^+= 40.5 (SCW^+= 0.5)).

4.3 Predicting Flow Rates through a T-Junction

Next, we attempt to predict the flow rates through the main and side channel based on a simple model that relies on the Bernoulli equation. Therefore, we first consider the (qualitative) behavior of the pressure along the centerline of the channels:

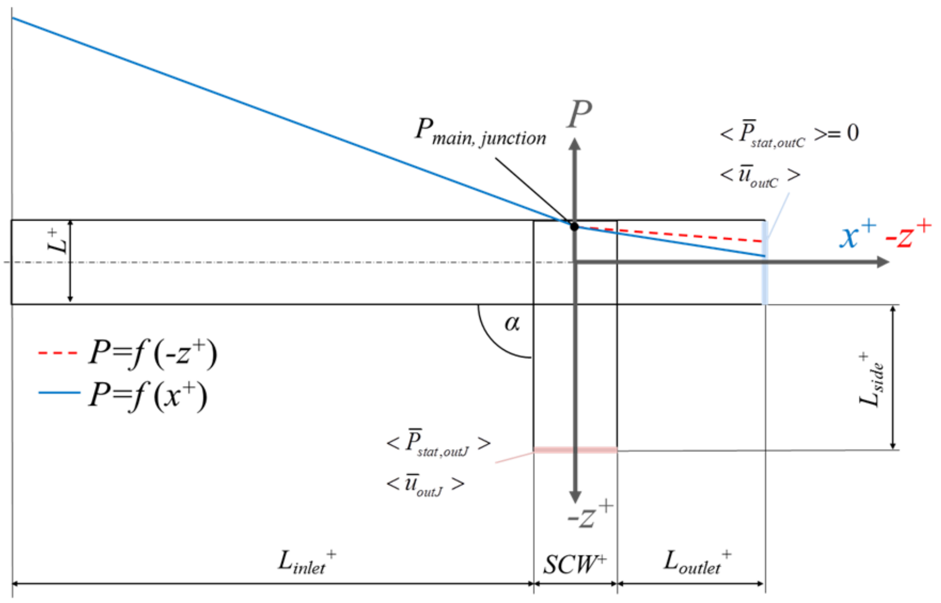


Fig. 22: Schematic representation of the pressure profiles near a T-junction.

Fig. 22 schematically indicates the profile of the mean pressure in the main (blue line) and the side channel (red, dashed line). In both channels the pressure decreases due to viscous dissipation. Most important, the pressure loss in the side channel is less than in the main channel due to the lower fluid velocity (note that in the base case the channel lengths have an equal length). In general it can be noted that the pressure loss in the inlet section of the main channel is the highest because of its length and the high velocity.

To build our model for the flow in a T-junction, we first summarize some key observations from the simulation results discussed in Chapter 4.1: first, it can be noticed from Fig. 18 that the volumetric flow fraction is greater than zero even if $\Delta p^+ \geq 0$. Clearly, this is because the pressure directly at the junction (i.e., $P_{main,junction}$) is somewhat higher than that at the main channel's outlet (i.e., $P_{stat,outC}$). Furthermore, for $\Delta p^+ = 0$, we observe from Fig. 18 that Φ^+ is substantially smaller than 1. Thus, a much larger amount of fluid exits through the main outlet (opposite of the inlet) compared to the side channel, even though the channel length and outlet pressure are identical. This indicates that the kinetic energy of the incoming fluid is pushing the fluid mainly towards the main outlet, and to a much lower extent to the side outlet. Thus, we need to consider the dynamic pressure contributing differently to the pressure distribution in the main and side channel. Another important fact is that the angle of the side channel has an influence on Φ^+. Hence, we expect that a second parameter (next to a friction factor) is necessary to model the split of the fluid streams at the T-junction. We do this by considering that the friction factor in the side channel is a factor ζ_J larger than that in the main channel. Furthermore, we allow ζ_J to vary with the angle of the side channel. The fact that $\zeta_J > 1$ can be explained by the fact that the flow in the side channel is confined to a small region next to the side channel's wall. Hence, the average shear stress acting on the walls of the side channel is higher, resulting in a higher pressure drop in this channel.

We are now in the position to model the pressure drop in the main and side channel using a simplified pressure loss calculation based on the well-known Bernoulli equation (see Appendix A.4.1 for details). Therefore, we assume that the pressure drop in the channel sections follows Hagen-Poiseuille's law, since the flow is expected to stay laminar. We then arrive at the following (implicit) model equation for the volumetric flow fraction:

$$\Phi^+ = \frac{4L^+ SCW^{+3}}{L_{outJ}\left(L^+ + SCW^+\right)^2 \zeta_J}\left(\frac{L_{outC}}{L^{+2}} - \frac{\Delta p^+ + \dfrac{<\overline{u}_{inC}>^2}{2}}{\zeta \dfrac{<\overline{u}_{inC}>}{\Phi^+ + 1}\dfrac{\mu}{\rho}}\right) \tag{3.1.1}$$

The only two fitting parameter in this equation are the loss coefficients ζ and ζ_J. The equation shows that $\Delta p^+ = 0$ does not imply $\Phi^+ = 0$, and that the outlet channel lengths play a significant role for the split of the flow. Furthermore, it can be seen that there is some critical (positive) pressure difference Δp^+ above which Φ^+ would become negative. For such a situation, inflow through the side channel would occur.

In order to determine the missing parameters, i.e., the loss coefficients, we next re-write the above equation to isolate the channels' main loss coefficient ζ:

$$\zeta = \frac{\Delta p^+ + \frac{<\bar{u}_{inC}>^2}{2}}{\frac{<\bar{u}_{inC}>}{\Phi+1}\frac{\mu}{\rho}} \frac{1}{\frac{L_{outC}}{L^{+2}} - \Phi^+ \frac{L_{outJ}\left(L^+ + SCW^+\right)^2}{4L^+ SCW^{+3}}\zeta_J} \tag{3.1.2}$$

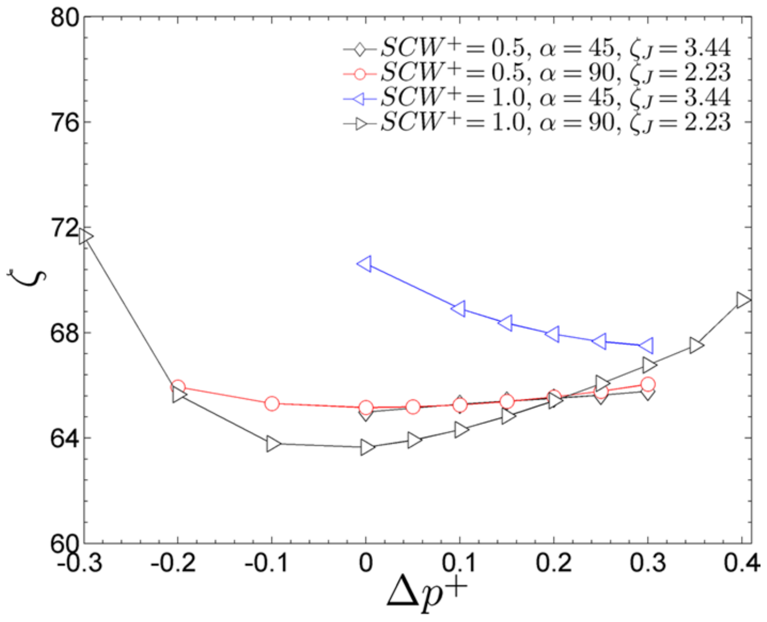


Fig. 23: Optimal loss coefficient for the main channel for different channel configurations (L_{inlet}^+=20, L_{oulet}^+=L_{side}^+= 6).

Doing so, allows us to compute ζ from our simulation results by assuming a certain value for ζ_J. By varying ζ_J in a way that yields (to a first approximation) a value for ζ independent of the applied pressure difference Δp^+, we arrive at Fig. 23. This figure summarizes the values for ζ when using the optimal value for ζ_J. Clearly, for the situation involving a narrow side channel (i.e., $SCW^+ = 0.5$) we observe an almost horizontal line close to $\zeta = 65.5$, indicating that our model assumptions are well satisfied for this situation. For $SCW^+ = 1.0$ we observe some variation of ζ in the order of approximately 10%. This indicates that our model assumptions are less accurate for such a situation, but still capture the main physics. Indeed, the comparison of the predicted and simulated values for Φ^+ assuming a fixed value for ζ shows good agreement for both channel widths (see Fig. 24). This demonstrates that our model detailed in Appendix A.4.1 is able to picture the key flow phenomena that govern the split of the flow at the T-junction.

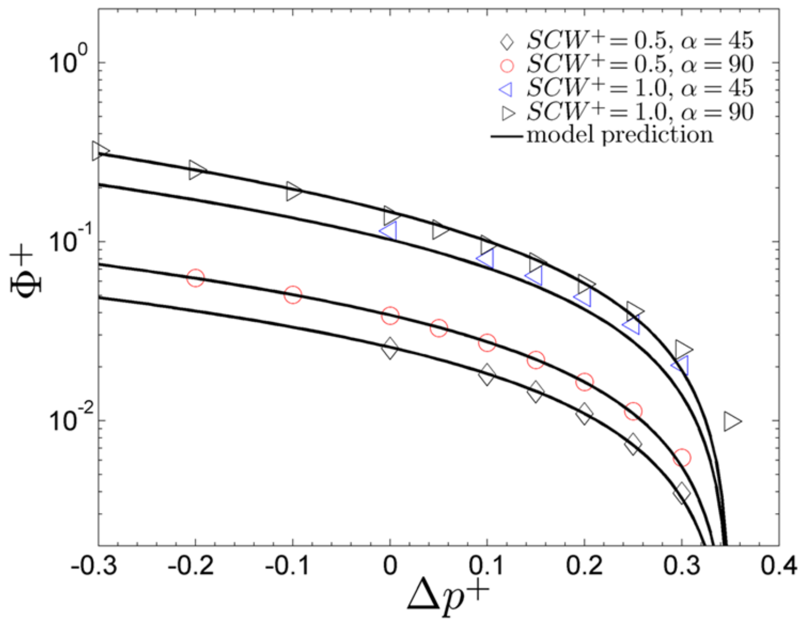


Fig. 24: Predicted versus simulated volumetric flow fraction at a T-junction (ζ = 65.5, ζ_J as per Fig. 23, L_{inlet}^+=20, L_{oulet}^+=L_{side}^+= 6).

5 Fibre-Fines Separation in a Dilute Suspension

The idea behind the fibre-fines separation in a T-junction is that the particles in the suspension rotate, and that fibres close enough to the wall touch the wall as a consequence of this rotation. Whenever a fibre contacts a wall it repels and moves towards the center of the channel. The position of the fibre's center of mass is then expected to be approximately half its length away from the wall (this is exactly true in case the fibre's major axis is initially perpendicular to the wall). Thus, longer fibres are more likely to reside near the center of the channel, in contrast to shorter fibres that are more homogeneously distributed across the channel. In summary, we expect the spatial distribution of suspended fibres depending on their length. This inhomogeneous distribution of particles can then be used to separate shorter fibres close to the wall by removing a thin layer of fluid (in which only small particles are present) via the side channel.

As a first step towards a more complete understanding of fibre-fines separation, a dilute suspension is considered here. In a dilute suspension the particles (i.e., fibres and fines) are isolated, and fibre-fibre interactions are neglected. Only fibre-wall interactions and hydrodynamic forces need to be accounted for, since they are needed to picture the separation mechanism proposed above. Based on this assumption, the T-junction dimensioning and the determination of ideal separation parameter settings are approached.

5.1 Explorative Studies

Before the actual separation simulation in a T-junction is considered, the general behavior of fibres in a laminar flow field, the impact of fibre-wall interactions, the extension of the critical separation layer, and the impact of the side channel on unseparated fibres is studied.

5.1.1 Fiber Behavior in a Straight Channel under Laminar Flow Conditions

The understanding and characterization of the actual behavior of a fibre (i.e., a prolate spheroid) in a laminar flow is the first step for the dimensioning of a hydrodynamic fractionation device. Jeffery [45] already described analytically the rotation of spheroids in linear, one-dimensional shear flow (see Appendix D). The numerical results gained from a simulation of fibre flow in a straight channel showed excellent agreement with Jeffery's analytical solution (the maximum error was as follows: *AR=2*: 4.5%, *AR=20*: 1.4%, for more details see Appendix D). Thus, the Jeffery orbit equation can be used for the calculation of the time period it takes for a fibre to flip.

We now assume that the fibre's velocity equals the fluid velocity at the fibre's center of mass, and that the flow is laminar and developed (see the analytical solution derived by Tamayol and Bahrami [46] summarized in Appendix D). The time period and the traveling length for a fibre to undergo a single Jeffery orbit at any position in the cross section can now be calculated. In case a fibre is sufficiently close to a wall (e.g., a distance closer than half its length in case the major axis of the fibre is oriented normal to the wall), less than a half orbit (i.e., a rotation of less than 180°) would be sufficient for the fibre to touch the wall. Subsequently, the fibre-wall interaction will change the fibre's position, such that under optimal conditions (i.e., a fibre oriented normal to the wall) the fibre's center of mass is located $d_{Major}/2$ away from the wall (see Fig. 25).

Note, the initial fibre orientation is a crucial factor for the repositioning of the fibre. In case a fibre is orientated parallel to the wall and perpendicular to the streamflow direction, the fibre tends to log-roll along the main channel which keeps the fibre from prepositioning and retards the separation efficiency.

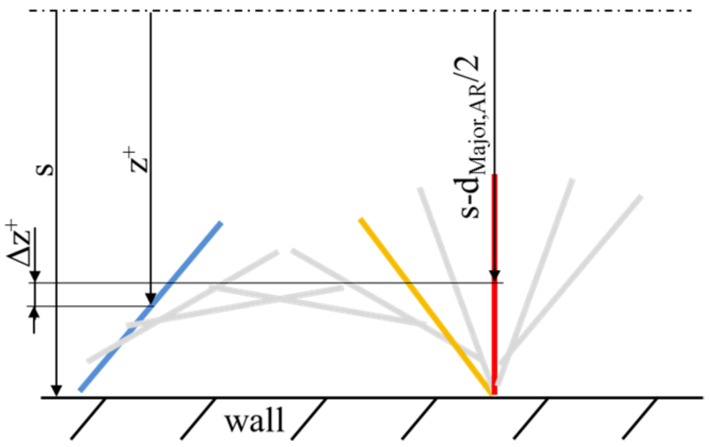


Fig. 25: Fibre repositioning due to a single fibre-wall interaction (flow is left to right). Blue line: initially positioned and oriented fibre. Grey lines(s): rotating fibre. Orange line: the fibre just contacts the wall. Red line: repositioned fibre located at a distance $s\text{-}d_{Major,AR}/2$ from the wall. The fibres center of mass moves Δz^+ towards the center of the channel due to a fibre-wall interaction.

Fig. 26 shows the analytically determined time period and traveling length needed for a half orbit for different fibres. The fibres are initially positioned at $y^+ = 0$ and at $s\text{-}d_{Major,AR=2}/2 \le z^+ \le s\text{-}d_{Major,AR=20}/2$. According to the analytical Jeffery orbit calculations, class $AR = 20$ (i.e., the class holding the longest fibres), requires the longest traveling length of approximately 18.45 dimensionless length units for a half orbit. The initial z^+-position of this slowest rotating fibre is $z^+ = s\text{-}d_{Major,AR=20}/2 = 0.3$ as shown in Fig. 26. For this reason the inlet length for the base configuration (for the T-junction simulation) is set to $L_{inlet}^+=20$.

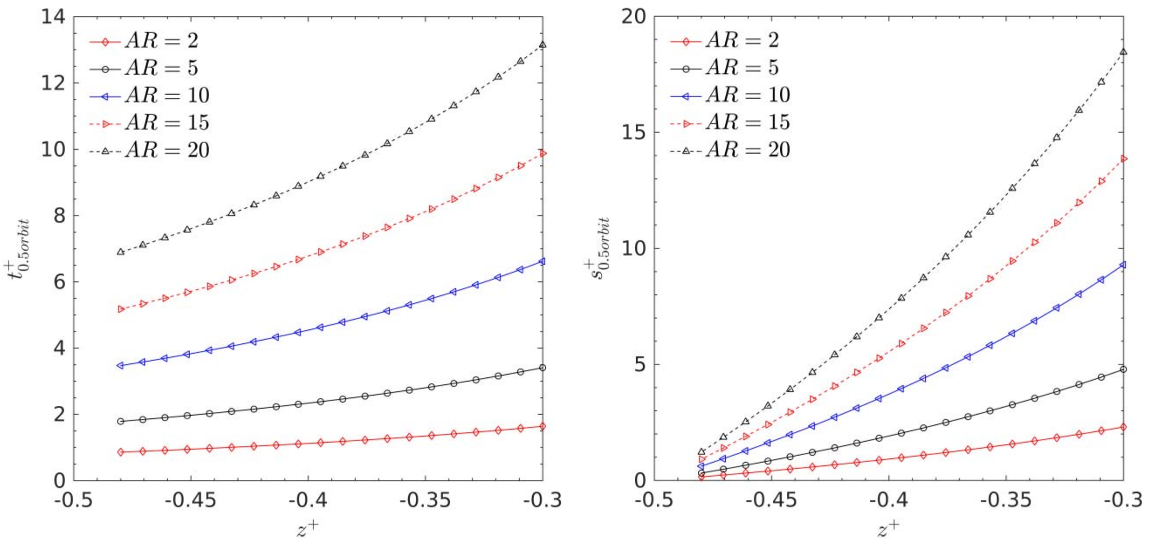


Fig. 26: Jeffery orbit calculations. Left Panel: time period of a half orbit $t^+_{0.5orbit}$. Right panel: traveling length of a half orbit $s^+_{0.5orbit}$.

5.1.2 Particle-Wall Impact

Now that the time period and the traveling length of a fibre, which possibly touches the wall, are known, the effect of a fibre-wall interaction is tested. Specifically, an idealized situation in which all fibres are initially oriented in streamwise direction (see Fig. 27) is considered. It can be shown that all fibres interacting with a wall for this hypothetical situation reposition themselves at a distance exactly half their length from the wall (see Fig. 28). This confirms our assumption that the removal of a thin suspension layer close to the wall implies a removal of short fibres (i.e., fines). This thin layer is called critical separation layer as suggested by literature [18], [19].

Again, we stress here that (i) the fibres must travel a sufficient distance to reposition themselves, as well as (ii) must be oriented exactly perpendicular to the wall at least once during their orbit.

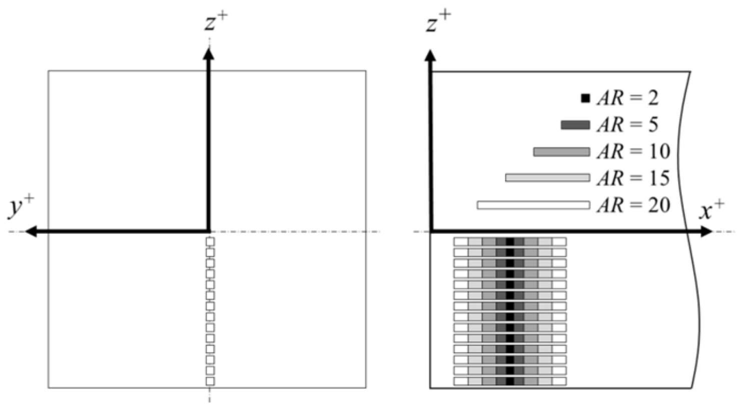


Fig. 27: Schematic representation of the ideal fibre position and orientation setup in a straight channel to investigate the impact of a fibre-wall interaction.

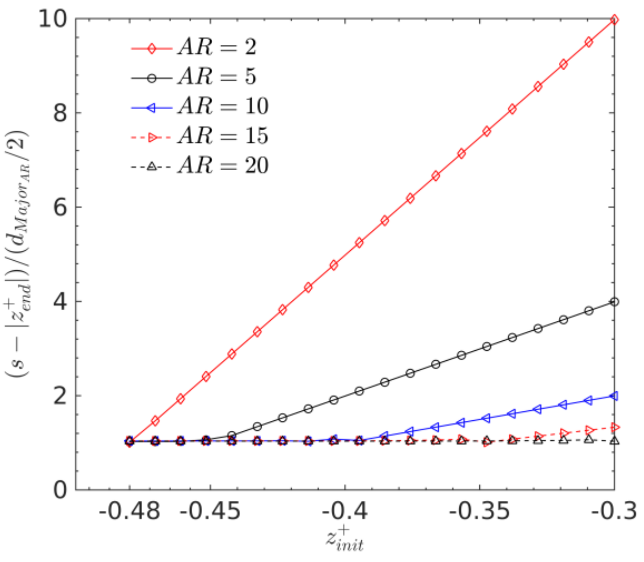


Fig. 28: The effect of fibre-wall interaction: fibres reposition due to wall contact at a distance z^+=*s*-$d_{Major,AR}$/2, i.e., (*s*-|z^+_{end}|)/($d_{Major,AR}$/2)=1 (initial fibre position and orientation as shown in Fig. 27).

5.1.3 Setup for the Analysis near a T-Junction

To analyze the separation process, three data collection planes (see Fig. 29) have been considered:

- the *preT*-plane is a y^+z^+-plane located at x^+=-0.51 ($L_{inlet}{}^+$=20) or x^+=-0.01 ($L_{inlet}{}^+$=40),
- the *postT*-plane is a y^+z^+-plane located at x^+=0.70, and
- the *sideC*-plane is a x^+y^+-plane located at z^+=-0.51.

The data of all fibres crossing any of these planes is collected. Thereby the position and orientation of all fibres before and after the T-junction section can be easily monitored. Also, it can be differentiated if a fibre gets separated the first time it moves over the T-junction, or later during the simulation.

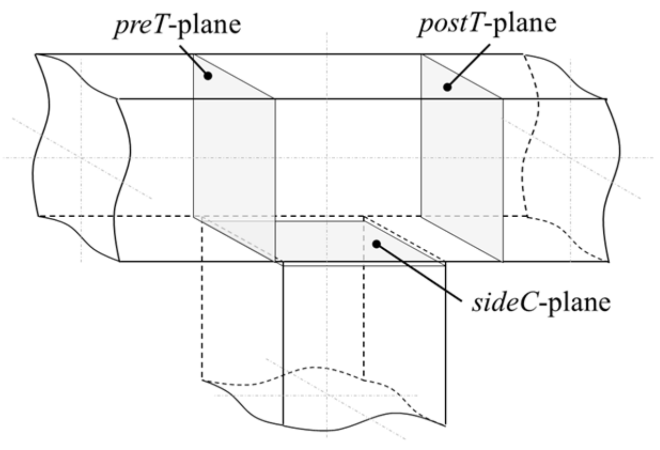
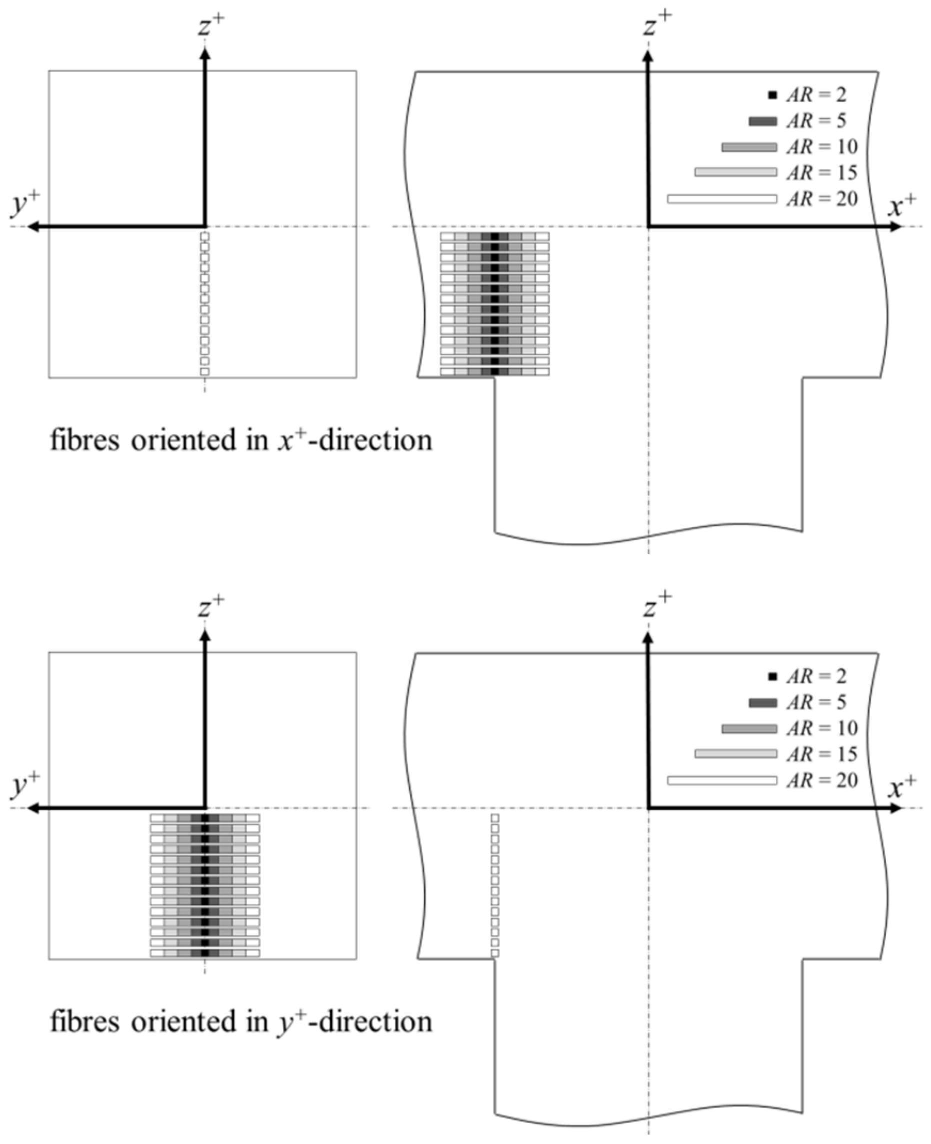


Fig. 29: Schematic representation of the data collection planes near the T-junction.

5.1.4 Critical Separation Layer

To determine the height of the critical separation layer, 100 fibres of each fibre class are placed just before the side channel (i.e., $x^+=-0.51$ for $SCW^+=1$; and $x^+=-0.01$ for $SCW^+=0.5$). Fibres are placed on a vertical line in the y^+z^+-plane ($y^+=0$, $-0.48 \leq z^+ \leq 0.0$), and are oriented in x^+- and y^+-direction as illustrated in Fig. 30. All fibres are assigned an initial velocity (in the streamwise direction) according to the local fluid velocity (see Appendix D for details on how the local fluid velocity can be calculated).



z^+
y^+
x^+
■ $AR = 2$
$AR = 5$
$AR = 10$
$AR = 15$
$AR = 20$
fibres oriented in x^+-direction
fibres oriented in y^+-direction

Fig. 30: Schematic representation of the initial fibre position and orientation in a T-junction to probe the position of the critical separation layer. Top: fibres oriented in x^+-direction. Bottom: fibres oriented in y^+-direction.

The fibre setup shown in Fig. 30 represents streamwise-oriented fibres, and fibres that will exhibit log-rolling [47]. These situations can be seen as the extreme orientations the fibres can have. The results of the determined height of the critical separation layers for all fluid flow cases (listed in Appendix A) is summarized in Fig. 31. The markers in this figure represent the separated fibre of each class with the highest initial z^+-position. Thus, the curves connecting these markers can be interpreted as the critical separation lines. The black horizontal line at $(s-|z^+_{max,sep}|)/(d_{Major}/2)=1$ indicates whether a separated fibres initial position is at a distance closer or further than half their length from the wall. In general it can be observed that the height of the critical separation layer is a function of the pressure difference, i.e., the volumetric flow fraction leaving through the side channel (see Chapter 4.1). The lower the volumetric flow in the side channel, the thinner the separation layer, and the lower the amount of long fibres that become separated. Regarding the initial orientation (streamwise directed or log-rolling) it can be seen that log-rolling fibres are more likely to get separated than streamwise directed fibres.

To isolate the most promising parameter settings out of these preliminary test cases, the ultimate goal of the separation, i.e., separation of only small particles, and the already gained knowledge about fibre behavior in laminar flow and the impact of fibre-wall interactions needs to be combined. For the actual separation at a T-junction all fibres have their starting position at the beginning of the main channel, and not immediately at the side channel as in the preliminary tests discussed above. Hence, fibres do have time, i.e., must travel a certain length through the channel, to interact with the wall and reposition themselves over the channel's cross section. Thus, for all separated fibres in these preliminary tests it can be (ideally) assumed that if their starting position is closer than half their length from the wall (below black line: $(s-|z^+_{max,sep}|)/(d_{Major}/2)<1$) they would not become separated if positioned at the beginning of the main channel. Thus, the criteria for usable parameter settings are (for both fibre orientations) (i) either only fibres from class *AR=2* are separated in these preliminary test cases, or (ii) that fibres from all classes become separated in these preliminary test cases but the fibres in larger classes (i.e., for *AR=5, 10, 15, 20*) hold a starting position closer than half their length from the wall (below black line).

In a first approach the cases revealing the most promising parameter settings (listed in Table 8) are found by investigating the streamwise-directed fibre results shown in Fig. 31 (left column) under the consideration of the above criteria. The results for the log-rolling fibre are not considered, since it is observed that fibres tend to be oriented more in streamwise direction.

Table 8: Cases with the most promising parameter settings.

case	SCW^+	α	Δp^+
SCW1.0_90_20_009	1.0	90°	+ 0.25
SCW1.0_90_20_010	1.0	90°	+ 0.30
SCW1.0_90_20_011	1.0	90°	+ 0.35
SCW1.0_45_20_005	1.0	45°	+ 0.25
SCW1.0_45_20_006	1.0	45°	+ 0.30
SCW0.5_90_20_007	0.5	90°	+ 0.20
SCW0.5_90_20_008	0.5	90°	+ 0.25
SCW0.5_90_20_009	0.5	90°	+ 0.30
SCW0.5_45_20_005	0.5	45°	+ 0.25
SCW0.5_45_20_006	0.5	45°	+ 0.30

Note, these separation lines, representing the height of the separation layers, are determined at ideal orientations (streamwise oriented and log-rolling fibres) and positions (y^+=0).

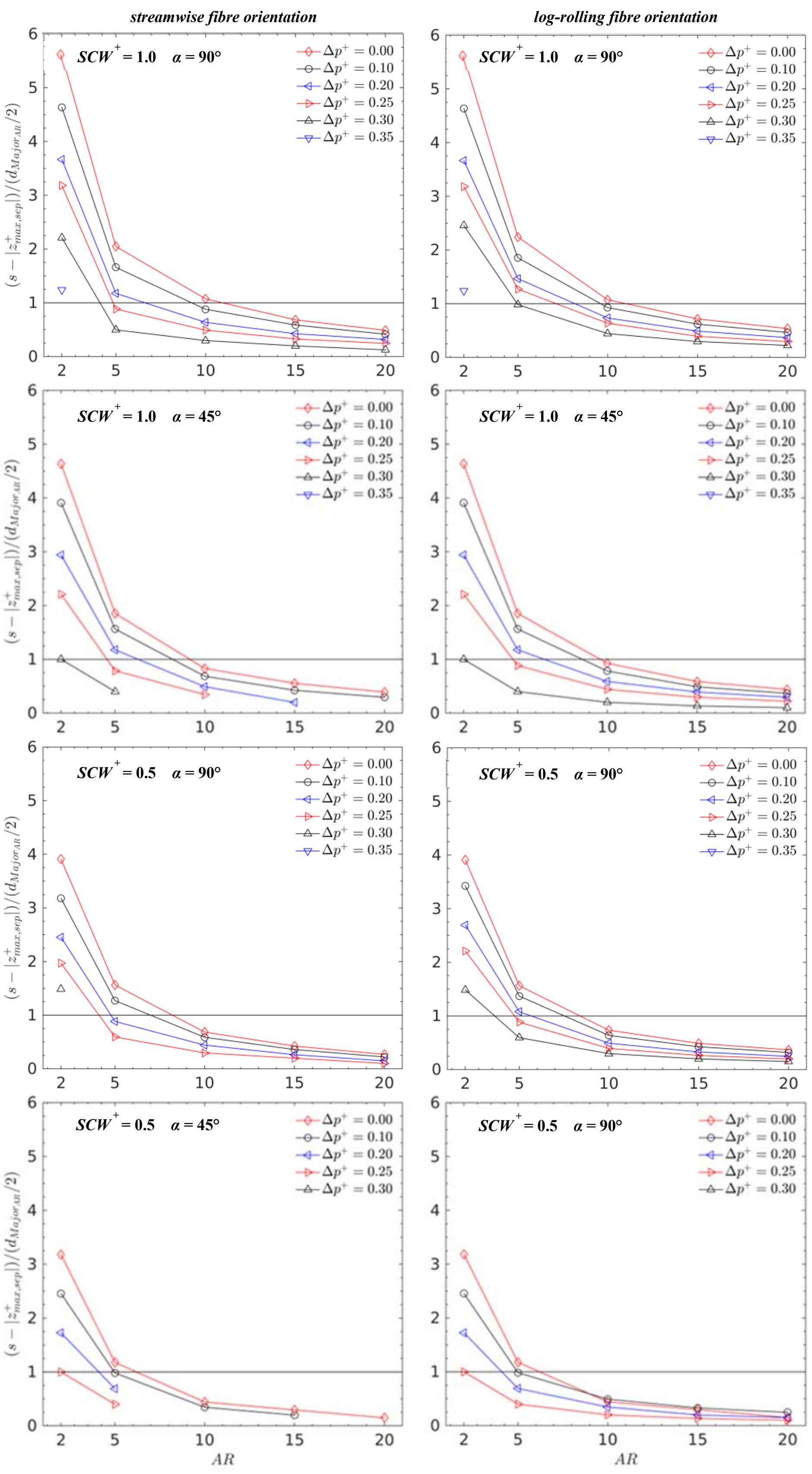


Fig. 31: Critical separation layer as a function of the fibres aspect ratio (left column: fibres initially oriented in the streamwise direction; right column: log-rolling fibres).

5.1.5 Impact of the Side Channel on Unseparated Fibres

While only a small portion of fibres becomes separated via the side channel, the remaining fibres are also influenced by the downwards suction of the side channel. It is now important to guarantee that small fibres that are not separated at the first junction (due to their large distance to the side channel) move slowly into the negative z^+-direction. By doing so, these particles will be separated at one of the following junctions. To quantify this behavior, fibres initially oriented in the x^+- and y^+-direction have been placed at $y^+ = 0$, similar to Fig. 30, and their trajectories have been followed.

Fig. 32 shows the change of the fibres' vertical distance when traveling from the *preT*-plane to the *postT*-plane. Unseparated fibres that are very close to the lower wall, i.e., the side channel entrance, either hit the wall, or the corner between the side channel and the main channel at the end of the T-junction. This results in a peak at the lower left corner of these graphs. Fibres initially oriented in the x^+-direction show an irregular curve progression also at a position very close to the wall opposite to the side channel. Those fibres rotate, touch the wall, repel and reposition themselves due to particle-wall interactions. Generally it can be noted that the downward suction only influences the fibres at the lower half of the cross section.

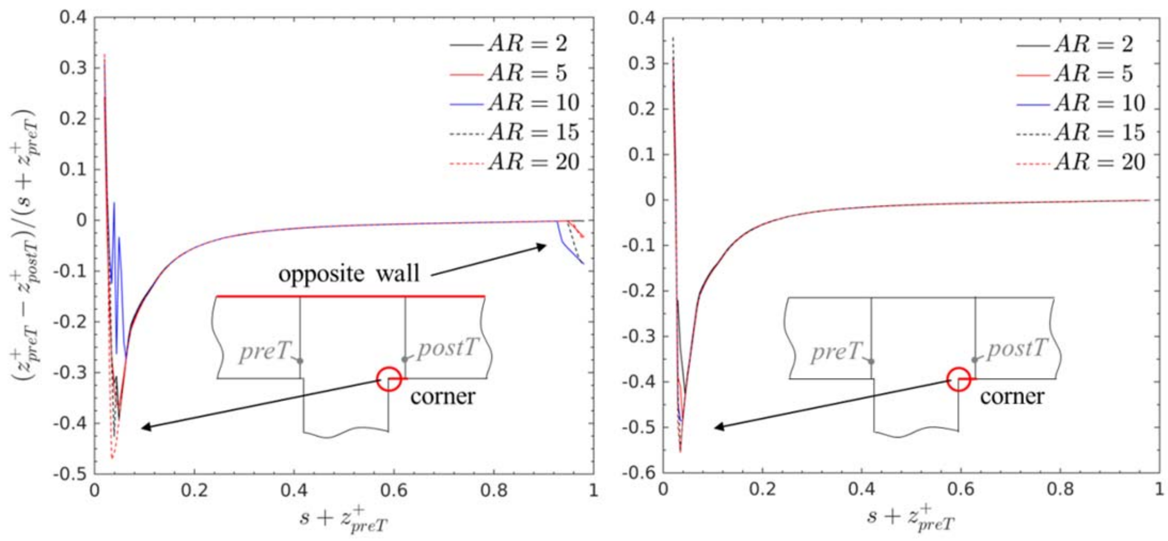


Fig. 32: Impact of the downwards suction of the side channel on the fibres remaining in the main channel. Left panel: fibres initially oriented in x^+-direction. Right panel: fibres initially oriented in y^+-direction. All fibres are initially positioned at $y^+ = 0$. The side channel entrance is located at $s+z_{preT}^+ = 0$, and the opposite wall is located at $s+z_{preT}^+ = 1$ (case: SCW^+=1, α=90°, Δp^+=0.35).

5.2 *Full Simulation Setup*

5.2.1 Fibre Position and Orientation at the Inlet

In the following simulations five fibre classes (with *AR=2, 5, 10, 15, 20)* containing á 200 fibres are used. The initial position of each fibre, its orientation and its velocity are assigned as explained below. Due to the symmetric fluid flow through the T-junction, it is sufficient to place the fibres at the beginning of the simulation only in one half of the cross section (i.e., at x^+=-20 for $L_{inlet}{}^+$=20 / ; x^+ = -40 for $L_{inlet}{}^+$=40, $y^+ \in$ [0 0.5], $z^+ \in$ [-0.5 0.5]). The distance from the wall is limited by half of each fibre class' length. Hence, shorter fibres are positioned on average closer to the wall than longer fibres. Fig. 33 shows the initial fibre position of all fibres in the y^+z^+-plane at the beginning of the main channel. The initial orientation is randomly chosen using LIGGGHTS®' *random quat* function, which provides a uniformly distributed random value for the fibres' quaternion (see Chapter C.2.5 for details). The initial fibre velocity was determined by using the velocity profile function for laminar flow in a square cross channel derived by Tamayol et al. [46] (see Appendix D).

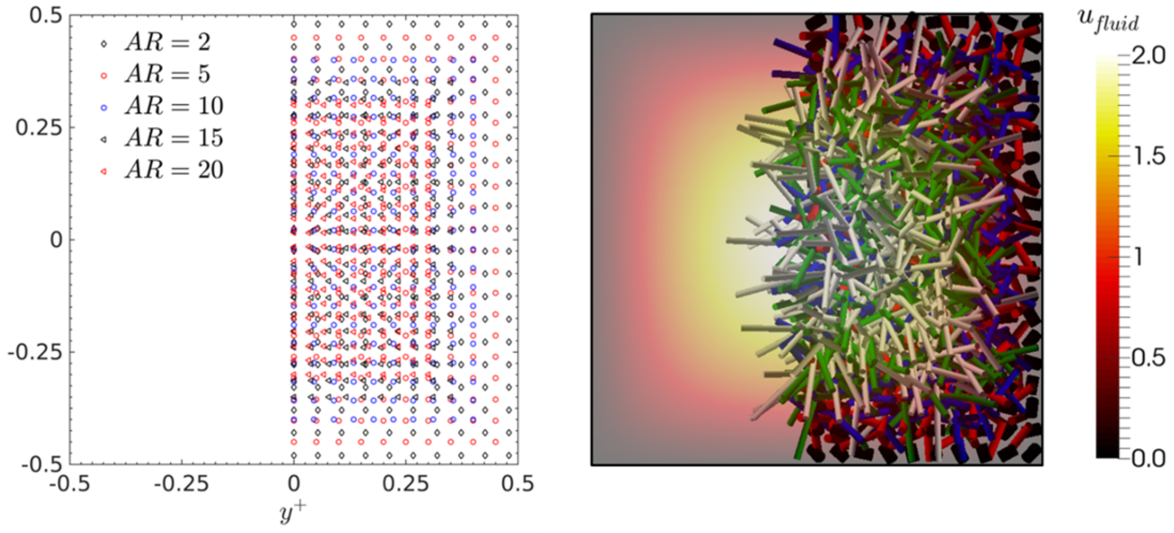


Fig. 33: Initial fibre positions in the y^+z^+-plane located at x^+ = -20 (for $L_{inlet}{}^+$=20) or x^+ = -40 (for $L_{inlet}{}^+$=40). Left panel: representation of each fibre's center of mass position. Right panel: fibres represented by cylinders in the channel.

5.2.2 Flow Setup

The CFD parameters in the CFD-DEM T-junction simulations are adopted from the CFD cases (see Chapter 3.1.3). Also the DEM parameters (e.g., the particle stiffness) are adopted from prior DEM simulations (see Table 6). For the coupling, a DEM time step of 10^{-6} is used with a coupling interval of 100 (i.e., 100 DEM time steps are performed during one update of the coupling forces).

The duration of all T-junction simulations with an inlet of L_{inlet}^{+}=20 is set to 116.35 dimensionless time units, the duration for all T-junction simulation with an inlet of L_{inlet}^{+}=40 is set to 229.85 dimensionless time units. Within this time the fibres closest to the wall and located at y^{+} = 0 (for which u_x is approximately 0.172) can move 20 (for L_{inlet}^{+}=20) or 40 (for L_{inlet}^{+}=40) length units, which equals the distance from their initial position to the end of the T-junction. Note that the slowest fibres that are located in the corners might not reach the junction during the simulation, since their velocities are more than ten times lower (i.e., u_x is approximately 0.013).

Furthermore, periodic boundaries for the particle phase present in the main channel are set: fibres leaving the main channel 0.5 dimensionless length units prior to the end of the main channel are re-injected at 0.5 dimensionless length units after the beginning of the main channel. This results in an effective distance between (the periodic) T-junctions of 26 dimensionless length units (for L_{inlet}^{+}=20; or 46 for L_{inlet}^{+}=40). Note, fibres positioned closer to the center of the channel travel with a higher velocity and may pass the T-junction multiple times. In contrast, fibres closer to the wall move with a smaller velocity and may pass the T-junction only once within the duration of the simulation (see Fig. 34). Hence, the actual number of serial T-junctions a fibre experiences is different for each fibre class. Unfortunately, this could not be avoided due to the limitation imposed by computational resources that avoided substantially longer simulations.

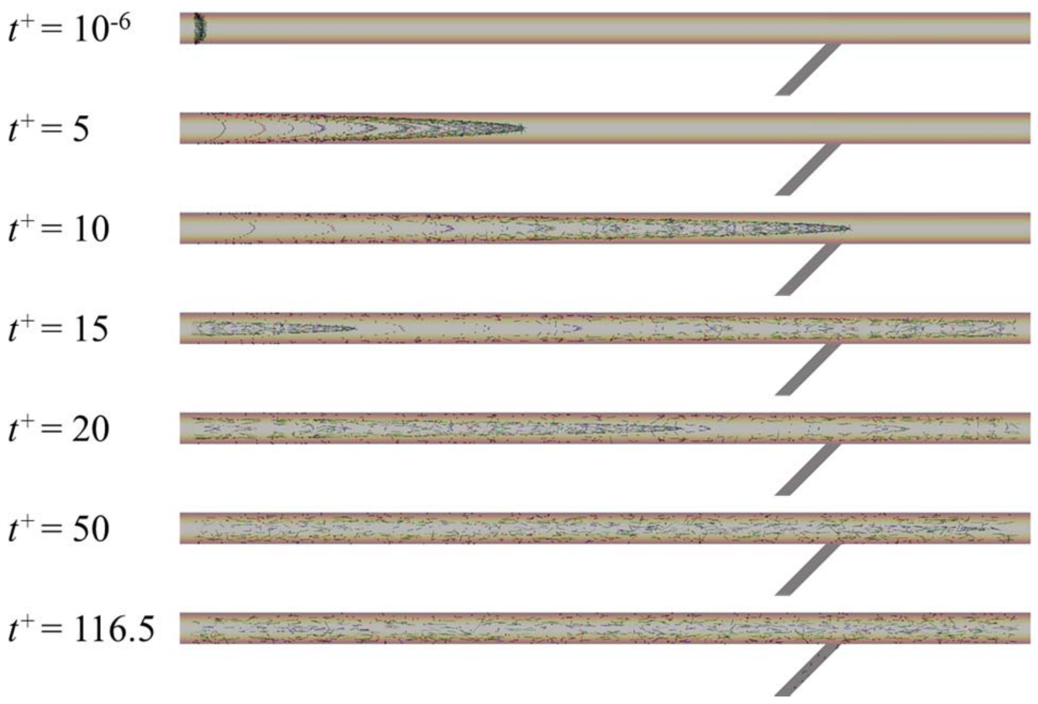


Fig. 34: Dilute fibre suspension flowing in T-junction at different instances in time. Fibres in the center of the main channel travel faster, and pass the junction more often than fibres closer to the walls.

5.3 Results for Fibre-Fines Separation near a T-Junction

5.3.1 Separation Efficiency

The separation efficiency [48] represents the fraction of each fibre class that is still in the main channel after passing a single, or multiple T-junctions. The base case parameters for the following analysis of the separation efficiency are: $Re = 500$, $L_{inlet}^{+} = 20$, $SCW^{+} = 1$, $\alpha = 90$. Based on this base case, parameter variations are performed and the results can be compared.

5.3.1.1 Comparison of the Results for Different Inlet Lengths

Since the first simulation performed with the base case parameters was not satisfying, it was attempted to influence the fibre orientation by doubling the inlet length. This provides the fibres a longer traveling distance to reposition themselves. As seen in Chapter 4.1, the inlet length has no influence on the volumetric flow fraction, thus it is clear that a longer inlet only influences the fibers' orientation but not the flow.

Fig. 35 shows the separation efficiency at the first T-junction, and over the whole simulation duration. As can be seen, an incomplete separation is observed for both inlet lengths. While at the first T-junction fibres from classes *AR=15* and *AR=20* are not separated, the separation efficiency of these classes decreases over the whole simulation duration. The length of the inlet mainly lowers the separation efficiency curves, but does not change the sharpness of the separation process. The smaller separation efficiency may results from the longer duration time in the $L_{inlet}^{+} = 40$ cases. Some fibres pass the junction section more often than in the $L_{inlet}^{+}=20$ cases. Thus, a direct comparison of the actual position of each curve is to some degree misleading. However, a qualitative observation of the curves leads to the conclusion that a larger traveling length does not positively influence the sharpness of cut. Thus, an inlet length of $L_{inlet}^{+}=20$ is kept constant for following cases.

The reason why a longer traveling length does not improve the sharpness of cut might be that the fibres that are oriented ideally in the streamwise direction reposition themselves anyway in the shorter channel. All other fibres oriented differently do not interact with the wall, even if they rotate their position is at the same distance from the wall than ideally oriented fibres. Some fibre orientation angles are just not favorable for a fibre-wall interaction, leading to no re-positioning of these fibres.

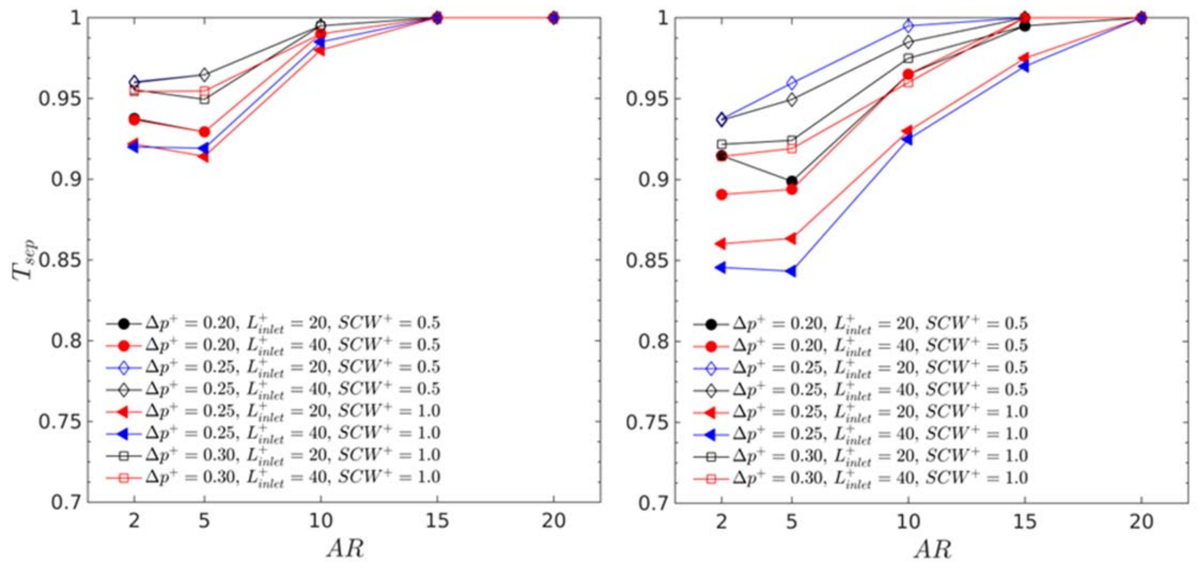


Fig. 35: Separation efficiency using different inlet lengths. Left panel: separation efficiency at first T-junction. Right panel: separation efficiency over whole simulation duration.

5.3.1.2 Results for the most Promising Parameter Settings

The most promising parameter settings discovered in the preliminary simulation (see Table 8) are now tested. Fig. 36 represents the separation efficiency of the first T-junction, as well as that recorded over the whole simulation duration. Again a decrease of the separation efficiency over the whole simulation can be observed. Furthermore, no case shows only a separation of fibres from class *AR=2*.

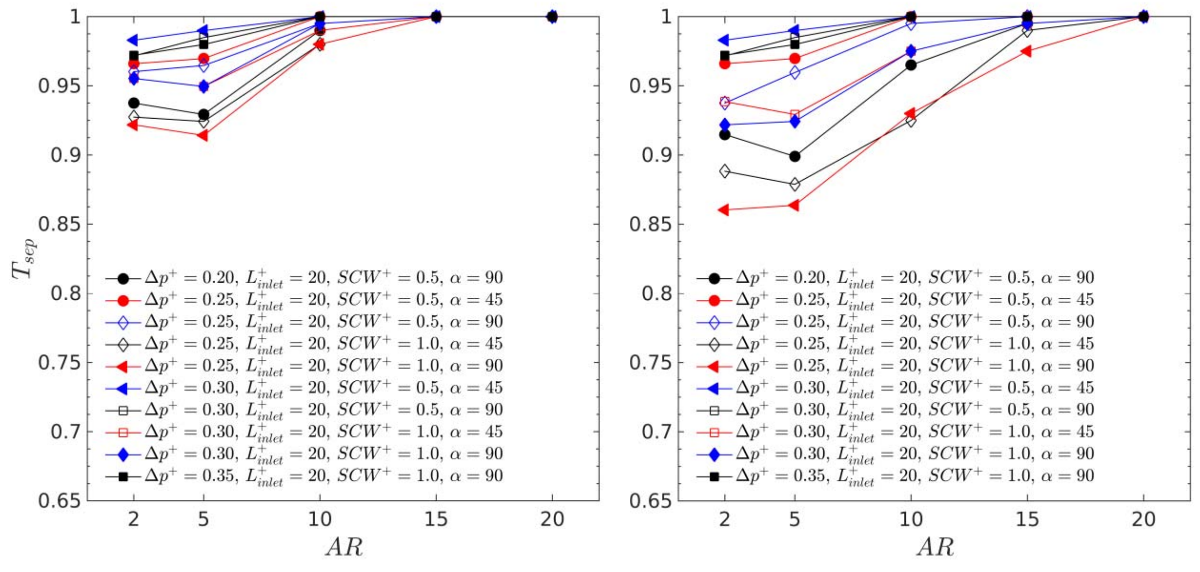


Fig. 36: Separation efficiency of the most promising parameter settings.

The best separation results, i.e., a separation of only fibres from *AR=2* and *AR=5* at the first and over the whole simulation duration, have been found in following cases:

Table 9: Cases with the best separation efficiency results.

Case	SCW^+	α	Δp^+
SCW1.0_90_20_011	1.0	90°	+0.35
SCW0.5_90_20_009	0.5	90°	+0.30
SCW0.5_45_20_005	0.5	45°	+0.25
SCW0.5_45_20_006	0.5	45°	+0.30

It seems that for all cases with an angle of α = 45° a side channel width of SCW^+ = 1 is too large, since for these cases (and different Δp^+ values) the separation efficiency curve is not satisfying, i.e., too flat.

The results for the separation efficiency for the best parameter settings are plotted in higher detail in Fig. 37. The separation efficiencies at the first T-junction do not differ from the separation efficiencies over the whole simulation. Assuming that the separation of fibres of class *AR=2* and *AR=5* is acceptable, case SCW0.5_45_20_005 with a side channel width of SCW^+ = 0.5, an angle α = 45°, and a pressure difference of Δp^+ = 0.25 seems to be the best choice to remove the fibres with the smallest number of T-junctions in a serial arrangement. Fig. 38 shows snapshots from the simulation of the optimal case (i.e., SCW0.5_45_20_005). These results indicate that only 2 to 4 % of the small particles are removed when considering one T-junctions. This suggests that a plurality (i.e., 50 or more) T-junctions are required to achieve a significant reduction of the fines content in a typical pulp.

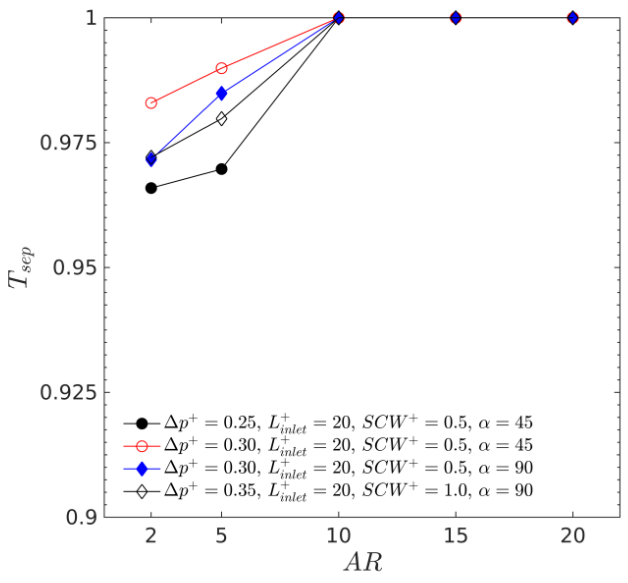


Fig. 37: Separation efficiency for best parameter settings.

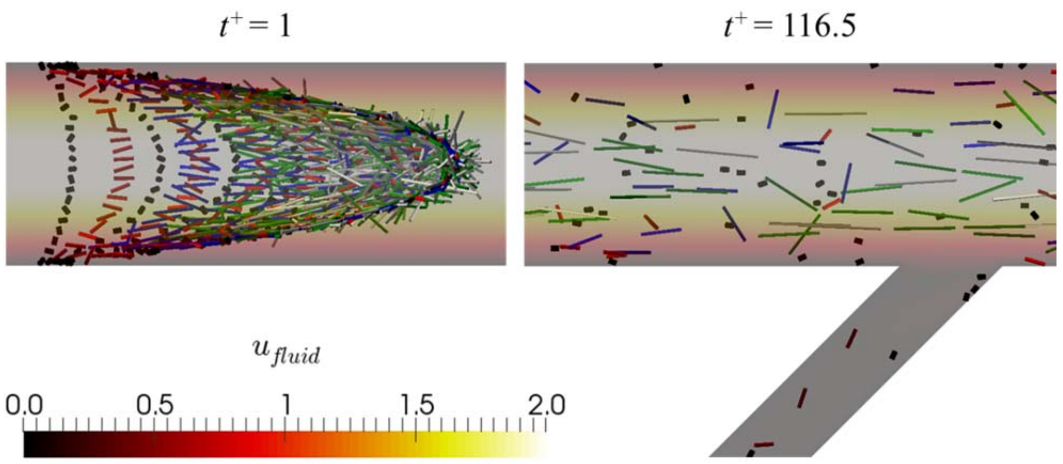


Fig. 38: Dilute fibre suspension in a channel with a T-junction at optimal operating conditions (case: SCW0.5_45_dp0.25). Left panel: fibres at the beginning of the simulation at the inlet (t^+=1). Right panel: T-junction, fibres from class *AR* = 2 and *AR* = 5 close to the bottom wall leave through the side channel (t^+ = 116.5).

5.3.1.3 Critical Separation Layer in the y^+z^+-Plane

The simulation results presented in this chapter differ from the expected results, i.e., that presented in Chapter 5.1. This has two main reasons:

- The critical separation line is not a straight line in the y^+z^+-plane, since the position of the fibres differ from $y^+ = 0$ at which the critical separation line was determined in preliminary tests.
- The orientation of the fibres is not always ideally in streamwise direction ($\theta = \pm 180°$, $\varphi = 90°$, i.e., the x^+-direction)

Thus, a more refined analysis is needed to quantify the exact topology of the critical separation layer in three-dimensional space. The critical separation line in the y^+z^+-plane is examined similar to the separation line in the x^+z^+-plane. Therefore, a plurality of fibres (of all classes) is placed in a C-shaped region in a plane whose normal is oriented in the x^+-direction (see Fig. 39). Furthermore, all fibres are oriented in the x^+-direction. Results are displayed in Fig. 40, in which it can be clearly observed that the separation line in the y^+z^+-plane is not straight: fibres near the corners of the channel can be further away from the side channel and still become separated. Surprisingly, this result is not affected by the shape of the fibres. This indicates that once the fibres have reached the frontal edge of the side channel, they follow the fluid's streamlines, and hence are separated independent of their shape. In summary, for the separation the fibres, also their lateral (i.e., y^+-) position is of crucial importance, and fibres must be re-positioned prior to the T-junction for an effective (i.e., shape-specific) separation.

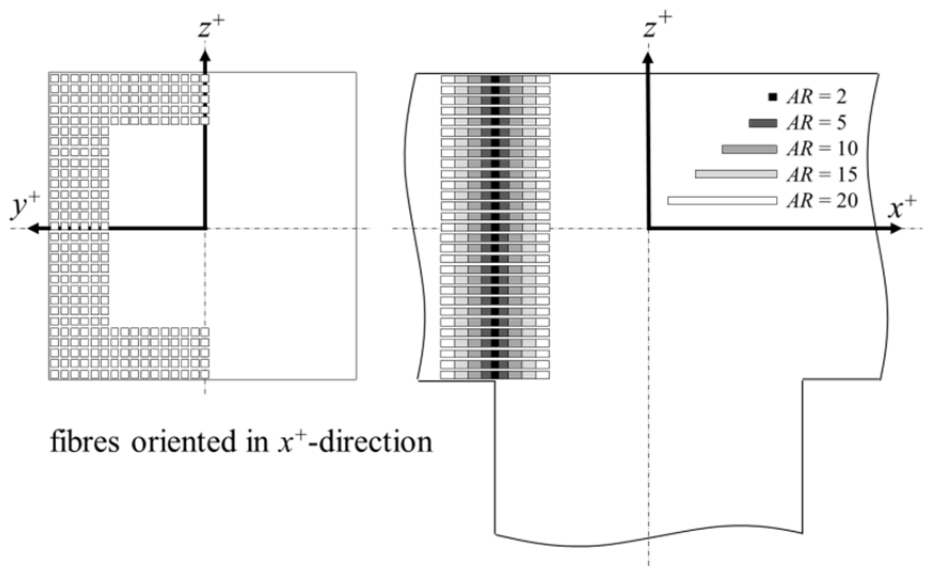


Fig. 39: Schematic representation of the simulation setup to study the extension of the critical separation layer. Fibres are positioned in a C-shaped region over the whole cross section, and are oriented in the x^+-direction.

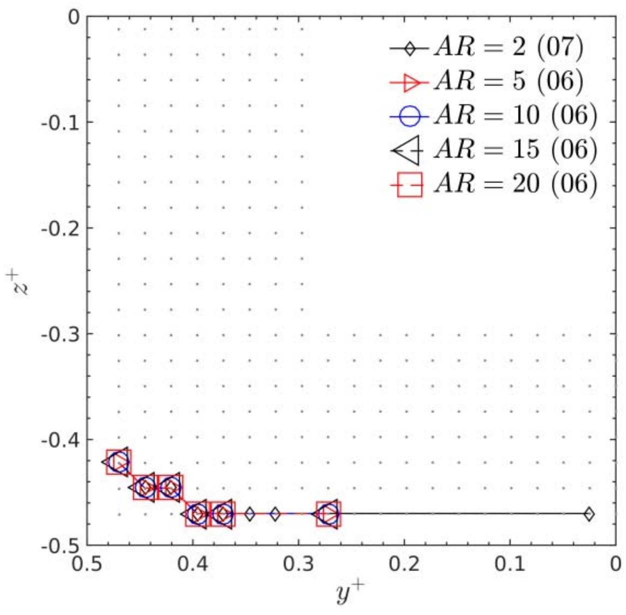


Fig. 40: Critical separation lines in the y^+z^+-plane for all fibre classes. Fibres are initially placed in the streamwise direction, and are injected just before the side channel at x^+=-0.51 (case: SCW^+=1, α=90°, Δp^+=0.35).

As already criticized, the fibres are oriented in either the x^+- or in y^+-direction in preliminary simulations. However, the fibre orientation will differ from those ideal cases in practice. Fig. 41 documents the orientation of all fibres passing the plane located at x^+ = -0.51 the first time. It can be seen that the fibres are more likely to be oriented in the x^+-direction, however, still differ from the streamwise-oriented fibers previously assumed.

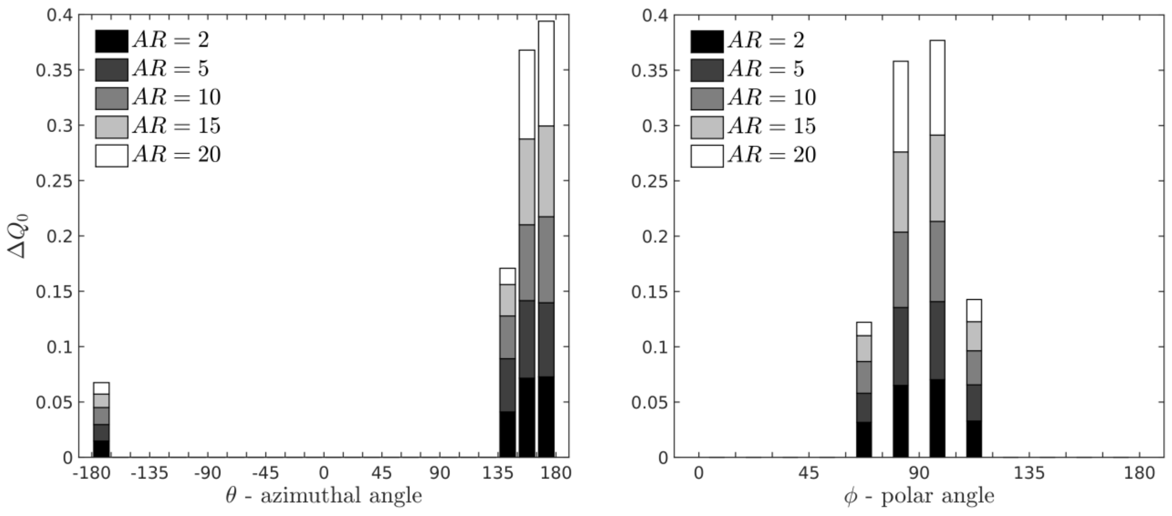


Fig. 41: Orientation of all fibres at x^+=-0.51 (case: SCW^+=1, α=90°, Δp^+=0.35), all classes have the same width of 15 degrees.

5.3.1.4 The Influence of Baffles

Now that the two main reasons for the deviation from an ideal separation are investigated, the idea of influencing at least one of these non-idealities seems to be logical. Hence, the orientation of the fibres should be favorably influenced, for example with comb-like internals, e.g., baffles near the inlet section of the channel. In the following considerations are made regarding the dimensioning of the baffles (see Fig. 42):

- The baffles have to be thin enough to not influence fluid flow
- The baffles distance has to be large enough so that very small fibres pass through and only the orientation of large fibres is influenced
- The position of multiple baffles has to be slightly shifted in the flow direction to avoid a simple upward rolling of a fibre on two (or more) baffles
- The inclination of the baffles has to be very small to avoid fibres to just bounce off, or form a rope that might plug the channel
- The maximum height of each baffle has to be kept low to avoid that fibres are just lifted in a region far away from the separation layer

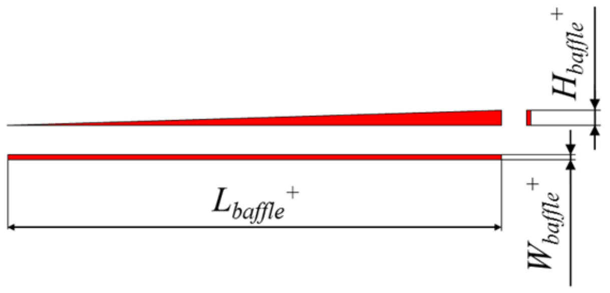


Fig. 42: Geometry of a single baffle.

The baffles' dimensions in the simulation and in the reference system are listed in Table 10. Note, the influence of the baffles on the fluid is neglected in the following simulations for simplicity.

Table 10: Baffle dimensions.

		L_{baffle}^{+}	H_{baffle}^{+}	W_{baffle}^{+}
simulation	[-]	10.0	0.1	0.02
reference system	[m]	$5.0 \cdot 10^{-2}$	$5.0 \cdot 10^{-4}$	$1.0 \cdot 10^{-4}$

The positions of the three shifted baffles are shown in Fig. 43. In all simulated cases the baffle parameters are *a* = 6, *b* = 11 and *c* = 16.

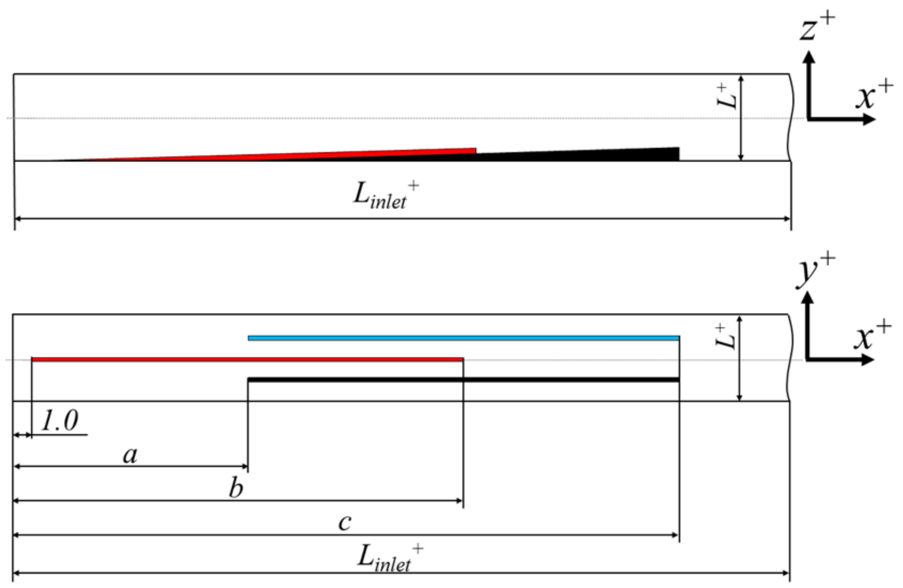


Fig. 43: Three shifted baffles in the inlet section of the channel. Top panel: side view of inlet section. Bottom panel: top view of inlet section.

The influence of the baffles is shown for the base case in Fig. 44. For the ideal parameter setting the baffles have no noticeable influence on the separation efficiency, which can be explained by the fact that there are no large fibres (*AR=10, 15, 20*) separated anyway. For the other cases it can be observed that the baffles influence the separation efficiency. The separation efficiency curves actually do lie higher if baffles are utilized. However, it cannot be seen that large fibres are completely excluded from separation, at least not with this baffle dimensions and positioning. Thus, a more rigorous analysis of the effect of baffles on the fibre orientation is needed to profoundly judge on their advantages or drawbacks.

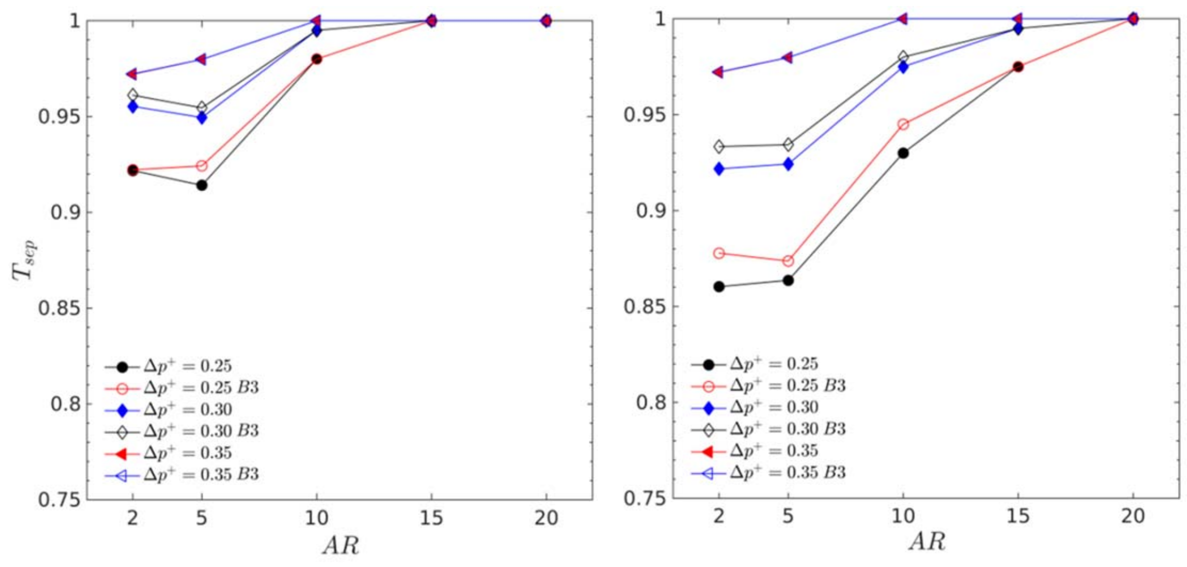


Fig. 44: Influence of baffles on the separation efficiency (case: L_{inlet}^+=20, SCW^+=1, α=90°, Δp^+=0.35). Left panel: at first T-junction. Right panel: over the whole simulation time.

5.4 Fibre Flock Behavior near a T-Junction

As a next step the previously neglected fibre-fibre interaction is applied in the fibre-fines separation in a T-junction. Thus the prior generated fibre flock (φ = 1.0%, see chapter 3.2.5) holding five fibre types with a distribution according to Table 5 is prepared to be inserted in the T-junction. The fibre flock is placed at 1.5 dimensionless length units after the beginning of the main channel, the DEM time step is set to 10^{-7} and the coupling interval to 100. Unfortunately no numerically stable setup could be found, neither by giving the fibres an initial velocity according to the fluid velocity profile [46], nor by reducing the DEM time step to a small, but realistic, time increment. It was found that in all simulations the potential energy at the beginning of the simulation increased rapidly, followed by the kinetic energy. The fibres in the fibre flock were ejected from the simulation, even against the flow direction. It is assumed that the fibre flock data stored after the fibre flock generation is not precisely enough (insufficient number of decimal places), and that this incorrectness leads to significant overlaps of fibres in the flock. Unfortunately, this occurs when the flock is loaded into the T-junction geometry. Thus, a work around strategy is used, by generating a fibre flock with 50% thicker fibres (larger d_{Minor} and d_{Major} values), but storing the input data for the T-junction simulation with the correct smaller values. Unfortunately, this strategy was also unsuccessful, it seems as if the reloading of the fibre flock in the channel geometry is not precise enough or fibres in the fibre flock are too entangled. Hence a closer investigation of the numerical problem, and maybe even a flexible fibre model are needed, as has been done in previous studies ([34], [35]).

6 Mechanism behind Fibre-Fines Separation

The mechanism behind the fibre-fines separation of a dilute suspension near a T-junction is mainly a combination of the height of the separated suspension layer, the fibre orientation and the fibre position. The separation layer removed by the side channel is directly influenced by the side channel dimensions, the angle between the side and the main channel, as well as the pressure difference set between the side channel's and the main channel's outlet. All fibres in the laminar flow field rotate, and their rotation rate increases with increasing distance from the center of the channel. Ideally, streamwise oriented fibres positioned close enough to the wall rotate, contact the wall, repel, and reposition themselves at exactly half their length from the wall. Differently oriented fibres might also touch the wall and reposition themselves, however, most likely at a position closer than half their length from the wall. Thus, a suboptimal fibre orientation leads to an unfavorable distribution of fibres over the cross section of the channel. For the fibre-fines separation it is preferred that only fibres from those fibre classes that need to be separated are within the region of the thin suspension layer removed by the side channel. Unfortunately, this cannot be always guaranteed due to a suboptimal fibre orientation. Furthermore, it should be considered that the height of this separated suspension layer varies over the cross section of the channel: near the corners of the side channel also fibres further away get drawn in, i.e., the critical separation line in the cross section is curved. Fibres that have the same distance to the wall, but are closer to a corner are more likely to get removed from the main channel. Hence, the fibre orientation is of high importance and should be the first parameter after Δp^+ that is influenced, e.g. by the usage of comb-like internals to align the fibres in streamwise direction.

7 Conclusions

7.1 Key Achievements

This thesis dealt with the virtual investigation of fibre-fines separation. OpenFOAM® simulations were performed to resolve the fluid flow in a T-junction geometry for different parameters. Due to symmetry problems, the meshing process was not only time consuming, but also led to a compromise to use a rather coarse mesh for the simulations. Thus, in this work only flows characterized by a rather low Reynolds number (Re=500) were investigated. One could criticize that the chosen mesh size is not fine enough to perform a DNS. However, here it is assumed that the error resulting from a (possible too) coarse mesh has no significant influence on key flow features, and thus has no significant influence on the predicted fibre-fines separation.

Subsequently, a fibre-fibre interaction model was introduced into the software tool LIGGGHTS® following the ideas of Lindström and Uesaka [31]. Subsequently, the model was used to generate a polydisperse fibre flock consisting of stiff fibres. The method used for the generation of the fibre-flock was a tri- or uni-axial deformation of a finite-sized cubical box. Fibres were initially randomly oriented in this box, and fluid flow was not considered. Such a dry method for flock preparation is not a realistic model for a flocculation process in reality, but is more a virtual trick to bypass the poorly characterized flocculation process [34]. The fibre flocks generated in this work are visually very similar to images taken from real stiff fibres (i.e., carbon fibres [49]). Unfortunately, fibres in the pulp and paper industry are often flexible [31], and thus the fibre flock generated virtually might differ in its properties from relevant fibre flocks used in the pulp and paper industry. However, the flock preparation strategy developed in this work is computationally rather efficient, and it was possible to investigate the effect of (virtual) flocculation conditions on the final fibre orientation statistics. The developed virtual flocculation processes can be easily adopted to study flexible fibres in the future.

Finally, optimal parameter settings for the fibre-fines separation at a T-junction were determined based on CFDEM® simulations. The strategy consisted of the following steps: (i) simulations regarding the behavior of fibres (e.g., fibre rotation rates) in a laminar flow in a straight channel, (ii) simulations to determine the height of the separation layer, and (iii) evaluation of the separation performance of a dilute fibre-fines suspensions (considering no fibre-fibre interactions) in a T-junction. As expected, the behavior of fibres in laminar flow showed perfect agreement with Jeffery's analytical solution [45] for fibre rotation. The newly developed correlation between side channel angle and the separation layer thickness confirmed previous experimental work (i.e., [18], [19], [50]) in which a critical separation line was proposed.

With the strategy developed in our work it was possible to find parameter settings that yield a perfect fibre-fines separation based on the particles' aspect ratio. Specifically, we showed that fines (i.e., particles with an aspect ratio smaller than five) were separated at the T-junction, while all long fibres followed the flow in the main channel. This translates to a sharp separation of fines with a length < 1mm in a typical industrial application. Most important, similar separation curves were found in laboratory experiments, even at higher pulp concentrations [44]. This suggests that our simulations can provide at least qualitatively correct information, and can guide the design process of hydrodynamic fractionation devices.

In conclusion, our key findings with respect to the optimal design of a flow fractionation device can be summarized as follows:

- a three-dimensional separation layer forms, which is influenced by the pressure situation in the system. The shape of this separation layer determines whether particles are separated or not. Hence, it is possible to find an optimal set of geometrical and process parameters that maximizes the sharpness of cut when separating particles with a length < 1 mm in an HDF.
- particle-wall interactions need to be taken into account, since they are the key separation mechanism in flow fractionation,
- baffles in the inlet channel have an effect on separation performance, and can potentially help to improve the sharpness of cut. However, a more rigorous analysis of the effect of baffles on the fibre orientation is needed to determine optimal baffle parameters.

7.2 Comparison with other Separation Technologies

A comparison to the most competitive separation device (i.e., a pressure screen) is summarized in Table 11. Pressure screens work based on the principle of surface filtration, i.e., the fact that fibres are simply too large to pass through an opening. In the case of pressure screens, the rotation of the fibres is induced by the rotor wing, which also re-suspends the fibre mat forming on the slot basket. In contrast, HDF devices work with fluid-flow induced segregation in the channel prior, and not at, the T-junction. The fibres' rotation is caused by the fluid flow, not by rotor, and hence expected to require a substantially lower amount of energy. In both devices the separation of the small particles is done via side channels (or slots).

Table 11: Comparison of hydrodynamic fractionation devices (HFD) versus pressure screens (PS, [12]).

	separation principle	**fibre rotation**	**device geometry**
HDF	**fluid flow induced segregation:** the suspension segregates due to wall-interactions before side channel, the non-homogeneous spatial distribution of fibres across the channel is used to only remove small particles	induced by fluid flow	serial T-junctions with variable angles and widths
PS	**surface area filtration:** small particles exit via slots in the screen. Fibres form a fibre mat on the slot basket, and are re-suspended via the rotor wing.	induced by rotor wing	slot basket with different profiles

7.3 Limitations and Outlook

In general it was assumed that for all CFD-DEM simulations the impact of the fluid on the fibre is sufficient enough if determined only at the center of mass of each fibre. This assumption might not be correct especially for longer fibres. Furthermore, the influence of the fibres on the fluid was neglected, since the mass loading of fibres is of order 0.01 or less in typical applications. Although recent studies attempted to introduce a fibre-fluid back-coupling strategy [51], we argue that including backcoupling would anyhow just be a first step to improve the predictions: when using unresolved CFD-DEM simulations, small-scale flow phenomena in the vicinity of the particles cannot be pictured directly, even when using backcoupling. Anyhow, addicting fibre-fluid back-coupling would be an interesting topic of future research.

The simulation of a fibre flock near a T-junction could not be performed due to numerical instability. It is believed that (i) the fibres are strongly entangled, i.e., a stress build-up in the flock occurs, and (ii) that the current strategy to save the orientation information of the fibres in the flock is not accurate enough to re-initialize the flock correctly in the T-junction geometry. In future it would be interesting to implement a flexible fibre model and to improve the data storage of the fibre orientation. This would enable us studying fibre-fines separation of a fibre flock under the influence of fibre-fibre interaction. Furthermore, a variation of the

Reynolds number, i.e., the flow regime, with the mapping technique introduced in chapter 3.1.4.2, as well as the usage of a finer mesh would be worth considering.

Finally, the effect of fibre roughness (i.e., the presence of fibrils on the surface of the fibres) should be investigated. Unfortunately, for a proper determination of the roughness length, it is expected that a dedicated experimental device is needed, which was not present at the time this thesis was written.

8 References

[1] M. Hermann, T. Pentek, and B. Otto, “Design Principles for Industry 4.0 Scenarios: A Literature Review,” 2015. [Online]. Available: http://www.leorobotics.nl/sites/leorobotics.nl/files/bestanden/2015 - Hermann Pentek & Otto - Design Principles for Industrie 4 Scenarios.pdf. [Accessed: 27-Oct-2015].

[2] M. Brettel, N. Friederichsen, M. Keller, and M. Rosenberg, “How Virtualization, Decentralization and Network Building Change the Manufacturing Landscape: An Industry 4.0 Perspective,” *Int. J. Mech. Aerospace, Ind. Mechatronics Eng.*, vol. 8, no. 1, pp. 37–44, 2014.

[3] A. Pfennig and E. Cvetkovic, *Model Development and Simulation Lecture Notes 2014*. Graz: CEET TU Graz, 2012.

[4] H. Plank, “Focused particle beam nano-machining: the next evolution step towards simulation aided process prediction,” *Nanotechnology*, vol. 26, no. 5, p. 050501, 2015.

[5] J. Chen, M. Zhang, Z. Yuan, and J. Wang, “Improved High-Yield Pulp Network and Paper Sheet Properties by the Addition of Fines,” *BioResources*, vol. 8, pp. 6309–6322, 2013.

[6] R. G. Tovar, W. J. Fischer, R. Eckhart, and W. Bauer, “White Water Recirculation Method as a Means to Evaluate the Influence of Fines on the Properties of Handsheets,” *BioResources*, vol. 10, no. 1953, pp. 7242–7251, 2015.

[7] Y. Xu and R. Pelton, “A New Look at How Fines Influence the Strength of Filled Papers,” *J. Pulp Pap. Sci.*, vol. 31, no. 3, pp. 147–152, 2005.

[8] E. Retulainen, K. Luukko, F. Kaarina, and J. Oere, “Papermaking Quality of Fines from different Pulps - the Effect of Size, Shape and Chemical Composition,” *55th Appita Annu. Conf.*, 2001.

[9] Andritz AG, “ModuScreen A: Screening and Fractionation,” 2010. [Online]. Available: http://www.andritz.com/pp-pulprecycled-finescreening-moduscreen-a.pdf. [Accessed: 27-Oct-2015].

[10] Andritz AG, “RotoWash: De-ashing and Fiber Recovery,” 2010. [Online]. Available: http://www.andritz.com/pp-pulprecycled-ashwashing-rotowash.pdf. [Accessed: 27-Oct-2015].

[11] Andritz AG, “SpeedWasher: De-ashing,” 2010. [Online]. Available: http://www.andritz.com/pp-pulprecycled-ashwashing-speedwasher.pdf. [Accessed: 27-Oct-2015].

[12] G. Lothar and H. Pakarinen, *Recycled Fiber and Deinking*, 7th ed. Helsinki, Finland: Fapet Oy, 2000.

[13] D. Di Carlo, “Inertial microfluidics.,” *Lab Chip*, vol. 9, no. 21, pp. 3038–3046, Nov. 2009.

[14] H. Schade and E. Kunz, *Strömungslehre*, 3rd ed. Berlin, Germany: Walter de Gruyter, 2007.

[15] H. Amini, W. Lee, and D. Di Carlo, “Inertial microfluidic physics.,” *Lab Chip*, vol. 14, pp. 2739–61, 2014.

[16] A. Lenshof and T. Laurell, “Continuous separation of cells and particles in microfluidic systems.,” *Chem. Soc. Rev.*, vol. 39, no. Nov. 2009, pp. 1203–1217, 2010.

[17] Z. Wu and K. Hjort, “Microfluidic Hydrodynamic Cell Separation: A Review,” *Micro Nanosyst.*, vol. 1, no. 3, pp. 181–192, Nov. 2009.

[18] S. Sugaya, M. Yamada, and M. Seki, “Observation of nonspherical particle behaviors for continuous shape-based separation using hydrodynamic filtration,” *Biomicrofluidics*, vol. 5, pp. 1–13, 2011.

[19] A. Tamura, S. Sugaya, M. Yamada, and M. Seki, “Tilted-Branch Hydrodynamic Filtration for Length-Dependent Sorting of Rod-Like Particles,” *Proc. 15th Int. Conf. Miniaturized Syst. Chem. Life Sci.*, pp. 1343–1345, 2011.

[20] J. Tu, G. Yeoh, and C. Liu, *Computational Fluid Dynamics: A Practical Approach*, 2nd ed. Oxford, UK: Elsevier, 2013.

[21] P. Moin and K. Mahesh, “Direct Numerical Simulation: A Tool in Turbulence Research,” *Annu. Rev. Fluid Mech.*, vol. 30, no. 1, pp. 539–578, 1998.

[22] S. W. Van Haren, “Testing DNS capability of OpenFOAM and STAR-CCM+,” Delft University of Technology, 2011.

[23] G. R. Tabor and M. H. Baba-Ahmadi, “Inlet Conditions for Large Eddy Simulation: A Review,” *Comput. Fluids*, vol. 39, no. 4, pp. 553–567, 2010.

[24] D. Vigolo, I. M. Griffiths, S. Radl, and H. a. Stone, “An experimental and theoretical investigation of particle–wall impacts in a T-junction,” *J. Fluid Mech.*, vol. 727, pp. 236–255, 2013.

[25] M. H. Baba-Ahmadi and G. Tabor, “Inlet Conditions for LES using Mapping and Feedback Control,” *Comput. Fluids*, vol. 38, no. 6, pp. 1299–1311, 2009.

[26] S. Luding, “Introduction to discrete element methods: Basic of contact force models and how to perform the micro-macro transition to continuum theory,” *Eur. J. Environ. Civ. Eng.*, no. Md, pp. 785–826, 2008.

[27] C. Marchioli, “COST FP1005 Trainings School May 27-29 2015. Behaviour of Regular and Irregular Non-Spherical Particles in Laminar and Turbulent Flows. Lecture Notes 1-3,” Haale(Saale), Germany, 2015.

[28] G. Lu, J. R. Third, and C. R. Müller, “Discrete Element Models for Non-Spherical Particle Systems: From Theoretical Developments to Applications,” *Chem. Eng. Sci.*, vol. 127, pp. 425–465, 2015.

[29] J. D. Redlinger-Pohn and S. Radl, “Paper in Preparation: Mechanistic Understanding of Fibre-Fines Separation in Coiled Tubes,” IPPT TU Graz, 2015.

[30] P. Schneider and Eberly David H., *Geometric Tools for Computer Graphics*. San Francisco, USA: Elsevier Science, 2003.

[31] S. B. Lindström and T. Uesaka, “Simulation of the motion of flexible fibers in viscous fluid flow,” *Phys. Fluids*, vol. 19, no. 11, 2007.

[32] A. N. J. Heyn, “The elementary fibril and supermolecular structure of cellulose in soft wood fiber.,” *J. Ultrastruct. Res.*, vol. 26, no. 1, pp. 52–68, 1969.

[33] J. Andrić, S. T. Fredriksson, S. B. Lindström, S. Sasic, and H. Nilsson, “A study of a flexible fiber model and its behavior in DNS of turbulent channel flow,” *Acta Mech.*, vol. 224, no. 10, pp. 2359–2374, 2013.

[34] C. Schmid and D. Klingenberg, “Mechanical Flocculation in Flowing Fiber Suspensions,” *Phys. Rev. Lett.*, vol. 84, no. 2, pp. 290–293, 2000.

[35] C. F. Schmid, L. H. Switzer, and D. J. Klingenberg, “Simulations of fiber flocculation: Effects of fiber properties and interfiber friction,” *J. Rheol. (N. Y. N. Y).*, vol. 44, no. 4, p. 781, 2000.

[36] R. G. Cox, “The motion of long slender bodies in a viscous fluid. Part 2. Shear flow,”

J. Fluid Mech., vol. 45, no. 04, pp. 625–657, 1971.

[37] Christopher J. Greenshields, “OpenFOAM The Open Source CFD Toolbox User Guide Version 2.4.0,” 2015. [Online]. Available: http://foam.sourceforge.net/docs/Guides-a4/UserGuide.pdf. [Accessed: 06-Jul-2015].

[38] R. Schwarze, *CFD-Modellierung*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[39] P. A. Cundall and O. D. L. Strack, “A discrete numerical model for granular assemblies,” *Géotechnique*, vol. 29, no. 1. pp. 47–65, 1979.

[40] M. Kvick, “Hydrodynamic stability and turbulence in fibre suspension flows,” Technical Reports from Royal Institute of Technology. KTH Mechanics Stockholm Sweden, 2012.

[41] Wikipedia, “Polar Coordinate System,” 2015. [Online]. Available: https://en.wikipedia.org/wiki/Spherical_coordinate_system#/media/File:3D_Spherical_2.svg. [Accessed: 06-Jul-2015].

[42] L. H. Switzer and D. J. Klingenberg, “Flocculation in simulations of sheared fiber suspensions,” *Int. J. Multiph. Flow*, vol. 30, no. 1, pp. 67–87, 2004.

[43] J. Andric, S. B. Lindström, S. Sasic, and H. Nilsson, “Numerical Investigation of Fiber Flocculation in the Air Flow of an Asymmetric Diffuser,” in *ASME 2014 12th International Conference on Nanochannels, Microchannels and Minichannels*, 2014, p. V001T12A013.

[44] J. König, “Master’s Thesis in Preparation: Experimental Study of Separation of Fines from Fiber Suspensions with Hydrodynamic Filtration,” IPPT TU Graz, 2015.

[45] S. Kim and S. J. Karrila, *Microhydrodynamics. Principles and Selected Applications*. Mineola, New York: Dover Publications, Inc., 2013.

[46] A. Tamayol and M. Bahrami, “Laminar Flow in Microchannels With Noncircular Cross Section,” *J. Fluids Eng.*, vol. 132, no. November, 2010.

[47] T. Rosén, F. Lundell, and C. K. Aidun, “Effect of fluid inertia on the dynamics and scaling of neutrally buoyant particles in shear flow,” *J. Fluid Mech.*, vol. 738, pp. 563–590, 2013.

[48] M. Stieß, *Mechanische Verfahrenstechnik - Partikeltechnologie 1*, 3rd ed. Springer-Verlag Berlin Heidelberg, 2009.

[49] Kreibe, “CFK Recycling: Der Stand der Dinge,” *Tech. Bayern*, vol. 5, p. 12ff, 2015.

[50] M. Yamada and M. Seki, “Hydrodynamic filtration for on-chip particle concentration and classification utilizing microfluidics.,” *Lab Chip*, vol. 5, pp. 1233–1239, 2005.

[51] J. Andric, “Numerical modeling of air – fiber flows,” Chalmers University of Technology Göteborg, Sweden, 2014.

[52] G. Junge, *Einführung in die Technische Strömungslehre*, 2nd ed. München: Fachbuchverlag Leipzig im Carl Hanser Verlag München, 2015.

[53] S. M. Salim and S. C. Cheah, “Wall y+ Strategy for Dealing with Wall-bounded Turbulent Flows,” *Int. MultiConference Eng. Comput. Sci.*, vol. II, 2009.

[54] S. Lyra, B. Wilde, H. Kolla, J. M. Seitzman, T. C. Lieuwen, and J. H. Chen, “Structure of hydrogen-rich transverse jets in a vitiated turbulent flow,” *Combust. Flame*, vol. 162, no. 4, pp. 1234–1248, 2015.

[55] T. Arens, H. Frank, Ch. Karpfinger, U. Kockelkorn, K. Lichtenegger, and H. Stachel, *Mathematik*, 2nd ed. Heidelberg: Spektrum Akademischer Verlag, 2010.

[56] C. Kloss, “LIGGGHTS(R)-Public Documentation, Version 3.X,” 2015. [Online]. Available: http://www.cfdem.com/media/DEM/docu/Manual.html. [Accessed: 28-Sep-2015].

[57] C. Goniva, C. Kloss, A. Hager, and S. Pirker, “An open source CFD-DEM perspective,” *Proceedings of OpenFOAM workshop*, 2010. [Online]. Available: http://ie.archive.ubuntu.com/disk1/disk1/download.sourceforge.net/pub/sourceforge/o/op/openfoam-extend/OpenFOAM_Workshops/OFW5_2010_Gothenburg/Papers/ChristophGonivaPaperOFWS5.pdf. [Accessed: 12-Aug-2015].

[58] D. Sunday, “Intersection of a Ray/Segment with a Plane,” 2001. [Online]. Available: http://geomalgorithms.com/a06-_intersect-2.html. [Accessed: 28-Sep-2015].

[59] H.-J. Bartsch, *Taschenbuch Mathematischer Formeln*, 21st ed. Munich, Germany: Carl Hanser Verlag, 2007.

[60] J. Wittenburg, H. A. Richard, J. Zierep, and K. Bühler, *Das Ingenieurwissen: Technische Mechanik*, 34th ed. Berlin Heidelberg: Springer-Verlag, 2014.

[61] E. Neumann, “Rigid Body Collisions,” 2003. [Online]. Available: http://www.myphysicslab.com/collision.html. [Accessed: 01-Jun-2015].

[62] E. W. Weisstein, “Spheroid,” *MathWorld-A Wolfram Web Resource*. [Online]. Available: http://mathworld.wolfram.com/Spheroid.html. [Accessed: 01-Jun-2015].

[63] T. Schwager and T. Poeschel, “Coefficient of restitution and linear dashpot model revisited,” *Granul. Matter*, vol. 9, no. 6, pp. 465–469, 2007.

[64] M. Otto, *Rechenmethoden für Studierende der Physik im ersten Jahr*. Heidelberg: Spektrum Akademischer Verlag, 2011.

[65] E. W. Weisstein, “Sphere Point Picking,” *MathWorld-A Wolfram Web Resource*, 2015. [Online]. Available: http://mathworld.wolfram.com/SpherePointPicking.html. [Accessed: 03-Jul-2015].

[66] S. Radl, “Direct Numerical Simulation of Reactive Deformable Bubbles in non-Newtonian Fluids,” IPPT TU Graz, 2006.

[67] C. Brennen, *Fundamentals of Multiphase Flows*. Pasadena, California: Cambridge University Press, 2005.

Appendix A CFD Simulation

A.1 Straight Channel

For an a priori investigation of the fibres' behavior in a laminar flow (i.e., the inlet channel of a T-junction) a CFD simulation in a straight channel geometry is performed. The cross section of the straight channel is identical to the cross section of the T-junction. The length of the straight channel $L_{straight}^{+}$ is 50 (see Fig. 45). A laminar inlet profile (see chapter 3.1.4.1) is used. For low Reynolds numbers (i.e., *Re*=500) no turbulences are expected thus the cell size in x^{+}-direction is kept low. The mesh is generated with the OpenFOAM® tool *blockMesh* with a total cell count of $1.2 \cdot 10^{5}$.

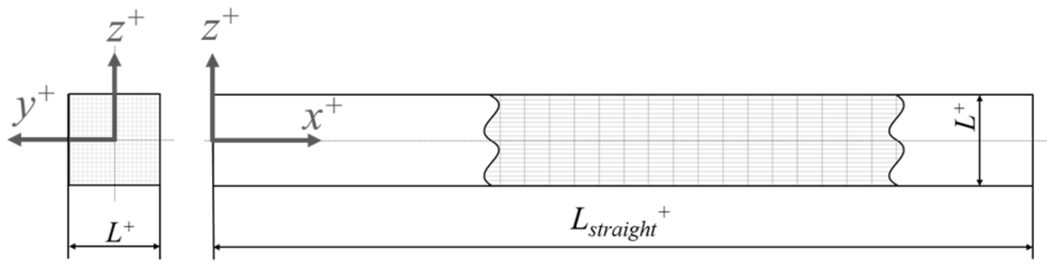


Fig. 45: Straight channel geometry. Grey mesh: quantitative representation of the mesh cells and their size.

In Table 12 and Table 13 the boundary conditions are summarized. The numerical schemes are identical to the schemes used for the T-junction (see chapter 3.1.3).

Table 12: Straight channel: fluid pressure boundary conditions.

patch	Type	inletValue		value	
inC	zeroGradient	-	-	-	-
outC	fixedValue	-	-	uniform	0
wall	zerogradient	-	-	-	-

Table 13: Straight channel: fluid velocity boundary conditions.

patch	Type	inletValue		value	
inC	fixedValue	-	-	uniform	(1 0 0)
outC	inletOutlet	uniform	(0 0 0)	uniform	(0 0 0)
		uniform	(0 0 0)	uniform	(0 0 0)
wall	fixedValue	-	-	uniform	(0 0 0)

A.2 Discretization Schemes [38]

To describe fluid motion mathematical formulations are needed. Those are mainly partial, non-linear differential equations based on the law of conservation for mass, momentum and energy, also known as Navier-Stokes Equations. Their general structure is:

$$\frac{\partial}{\partial t}(\rho\phi)+\nabla\cdot(\rho\mathbf{u}\phi)=\nabla\cdot(\Gamma\nabla\phi)+Q_\phi \tag{A.2.1}$$

ρ is the density of the fluid, t stands for the time, ϕ represents any fluid parameter (e.g. fluid velocity **u** or temperature *T*), Γ is the diffusion coefficient and Q_ϕ represents sources and sinks (e.g., through particle-fluid interactions). The first term describes the change of ϕ over time, the second term stands for convective flux, the third for diffusive fluxes and the last term represents any sources and sinks. Through the specification of each term the three conservation laws can be generally described as:

- mass conservation: $$\frac{d\rho}{dt}=\frac{\partial\rho}{\partial t}+\nabla\cdot(\rho\mathbf{u})=0 \tag{A.2.2}$$
- momentum conservation: $$\frac{d}{dt}(\rho\mathbf{u})=\frac{\partial}{\partial t}(\rho\mathbf{u})+\nabla\cdot(\rho\mathbf{u}\mathbf{u})=\nabla\cdot\boldsymbol{\tau}-\nabla p+\rho\mathbf{g} \tag{A.2.3}$$
- energy conservation: $$\frac{\partial}{\partial t}(\rho h)+\nabla\cdot(\rho\mathbf{u}h)=-\nabla\cdot\mathbf{q}+\frac{\partial p}{\partial t}+\nabla\cdot(\boldsymbol{\tau}\cdot\mathbf{u}) \tag{A.2.4}$$

τ represents the shear stress tensor, p the pressure, g the gravity, h stands for the enthalpy and q is the heat flux.

A.3 Pressure Loss

To determine the pressure loss, the Bernoulli equation:

$$\frac{\Delta P_{loss}}{\rho_{Fluid}}=\frac{<\overline{P}_{stat,inC}>}{\rho_{Fluid}}-\frac{<\overline{P}_{stat,outC}>}{\rho_{Fluid}}+\frac{<\overline{u}_{inC}>^2}{2}-\frac{<\overline{u}_{outC}>^2}{2} \tag{A.3.1}$$

was applied [52]. The data needed for this equation, i.e., $<\overline{P}_{stat,i}>/\rho_{fluid}$ and $<\bar{u}_i>$ at the inlet and the outlet of the main channel, can be extracted from the OpenFOAM® simulations. Note, that time- and spatial averaging over the inlet and outlet plane was applied.

A.4 Volumetric Flow Fraction

The volumetric flow fraction Φ^+ is defined as:

$$\Phi^+ = \frac{<\dot{V}_{outJ}>}{<\dot{V}_{outC}>} = \frac{<\bar{u}_{outJ}> A_{outJ}}{<\bar{u}_{outC}> A_{outC}} = \frac{<\bar{u}_{outJ}> L^+ SCW^+}{<\bar{u}_{outC}> L^{+^2}} \tag{A.4.1}$$

The time-averaged velocities $<\bar{u}_i>$ are averaged over the outlet planes out_C and out_J, and the multiplied with each plane's cross section area A_i.

A.4.1 Relationship between Pressure and Volumetric Flow Fraction

As seen in Fig. 18, the volumetric flow fraction Φ^+ is larger than zero if $<\bar{P}_{stat,\mathrm{outJ}}> = 0$. To model the correlation between volumetric flow fraction and the pressure at the outlets, the following assumptions are made:

- pressure drop $\Delta p \propto \frac{<\bar{u}_i>^n L \mu^k \rho^r}{H^m}$ (yielding the Hagen–Poiseuille equation [14] for $m = 2$, $n = 1$, $k = 1$, $r = 0$, or a quadratic pressure loss typical for turbulent flow in case $m = 1$, $n = 2$, $k = 0$, $r = 1$)
- the loss coefficient in the main channel is $\zeta_{channel} = \zeta$
- the loss coefficient in the side channel is $\zeta_J^{'} = \zeta_J \zeta$

Here $<\bar{u}_i>$ indicates time-averaged velocities at one of the two outlets, L stands for the length of a straight channel part, μ is the dynamic viscosity, ρ is the fluid density, and H is the hydraulic diameter. The hydraulic diameter is defined as $H=4A/U$ [14], where A is the crosssectional area, and U is the perimeter of the cross section of the main channel (index "C"), or the side channel (index "J").

$$\begin{aligned} H_c &= \frac{4A}{U} = \frac{4L^+L^+}{4L^+} = L^+ \\ H_j &= \frac{4A}{U} = \frac{4L^+SCW^+}{2\left(L^+ + SCW^+\right)} = \frac{2L^+SCW^+}{L^+ + SCW^+} \end{aligned} \tag{A.4.2}$$

The pressure at the crossing point of the main and side channel can now be defined with the usage of the generalized pressure drop model introduced above. Specifically, a Bernoulli equation [52] is established in the following from (i) the beginning of the outlet section of the main channel to the end of the outlet section, and (ii) from the beginning of the side channel to the side channel outlet.

Our simulations indicate an uneven split of the flow in case no pressure difference is applied to the channel outlets. Thus, we need to consider the dynamic pressure (associated with the incoming flow) acting differently on the two outlet channels. Specifically, we use two indicator variables, i.e., π_m, π_J, to account for this.

Finally, the Bernoulli equation for the main and side channel is, respectively:

$$\frac{\overline{P}_{main,junction}}{\rho_{Fluid}} + \pi_m \frac{<\overline{u}_{inC}>^2}{2} = \underbrace{\frac{<\overline{P}_{stat,outC}>}{\rho_{Fluid}}}_{=0} + \frac{<\overline{u}_{outC}>^2}{2} + \frac{<\overline{u}_{outC}>^n L_{outC}}{L^{+m}} \zeta \; \mu^k \rho^{r-1} \tag{A.4.3}$$

$$\frac{\overline{P}_{main,junction}}{\rho_{Fluid}} + \pi_J \frac{<\overline{u}_{inC}>^2}{2} = \underbrace{\Delta p^+}_{\frac{<\overline{P}_{stat,outJ}>}{\rho_{Fluid}}} + \frac{<\overline{u}_{outJ}>^2}{2} + \frac{<\overline{u}_{outJ}>^n L_{outJ} \left(L^+ + SCW^+\right)^m}{\left(2L^+ SCW^+\right)^m} \zeta_J \zeta \; \mu^k \rho^{r-1} \tag{A.4.4}$$

By combining equations (A.4.3) and (A.4.4) we arrive at:

$$\Delta p^+ = \left(\frac{<\overline{u}_{outC}>^n L_{outC}}{L^{+^m}} - \frac{<\overline{u}_{outJ}>^n L_{outJ} \left(L^+ + SCW^+\right)^m \zeta_J}{\left(2L^+ SCW^+\right)^m} \right) \zeta \; \mu^k \rho^{r-1} + \frac{<\overline{u}_{inC}>^2}{2} \left(\pi_J - \pi_m\right) \tag{A.4.5}$$

We can now isolate the loss coefficient ζ associated with the main channel:

$$\zeta = \frac{\Delta p^+ - \frac{<\overline{u}_{inC}>^2}{2}\left(\pi_J - \pi_m\right)}{\left(\frac{<\overline{u}_{inC}>}{\Phi+1}\right)^n \mu^k \rho^{r-1}} \; \frac{1}{\frac{L_{outC}}{L^{+m}} - \Phi^{+n} \frac{L_{outJ}\left(L^+ + SCW^+\right)^m}{2^m L^{+(m-n)} SCW^{+(m+n)}} \zeta_J} . \tag{A.4.6}$$

Similarly, the volumetric flow fraction (note, that this is an implicit relationship) can be extracted:

$$\Phi^+ = \sqrt[n]{\frac{2^m L^{+(m-n)} SCW^{+(m+n)}}{L_{outJ}\left(L^+ + SCW^+\right)^m \zeta_J}} \sqrt[n]{\left(\frac{L_{outC}}{L^{+m}} - \frac{\Delta p^+ - \frac{<\overline{u}_{inC}>^2}{2}\left(\pi_J - \pi_m\right)}{\left(\frac{<\overline{u}_{inC}>}{\Phi^+ + 1}\right)^n \mu^k \rho^{r-1} \zeta} \right)} \tag{A.4.7}$$

These relationships explains why the volumetric flow fraction Φ^+ can be greater than zero even if $< \overline{P}_{stat,\mathrm{outJ}} > =0$, as well as helps to understand the effect of the outlet channel length on the split of the flow. By fitting the parameters ζ and ζ_J to simulation (or experimental) data, it is now possible to calculate Φ^+ for any combination of geometrical and operating parameters of the channel.

A explorative comparison of the simulation data with predictions made with the above model indicates that choosing $\pi_J = 0$, $\pi_m = 1$, and assuming a pressure drop typical for laminar flow, yields the best agreement between simplified model and detailed simulations.

A.5 Dimensionless Wall Distance y^+

If DNS is used, it is important that the mesh grid, which is used in the simulation, is fine enough to resolve the whole flow, especially in wall regions, where large gradients in the solution appear due to viscous effects, the mesh grid might need to be finer. Generally the near-wall region of large gradients is divided into layers and the description of the size of each of these layers is done by the dimensionless wall distance y^+:

$$y^+ = \frac{u_\tau y}{\nu} \qquad u_\tau = \sqrt{\frac{\tau_w}{\rho}} \tag{A.5.1}$$

with u_τ the friction velocity, y the normal distance from the wall, ν the kinematic viscosity of the fluid, τ_W the wall shear stress and ρ the fluid density [20]. The near-wall region is mainly divided in viscous sublayer $y^+ < 5$, buffer layer $5 < y^+ < 30$ and fully turbulent/log-region layer $y^+ > 30$ to 60 [53]. In DNS the rule of thumb for a fine enough mesh grid near the wall is $y^+ < 1$ [54].

A.6 CFD Benchmark Simulation Data

A.6.1 Detailed Case Data

Following Tables hold all CFD case settings and results.

Table 14: CFD Cases Simulation Data Part 1.

Case	#cells	*Re*	L_{outlet}^+	L_{intlet}^+	SCW^+	α	Δp^+	$t^+_{ave,start}$	t^+_{ave}	t^+_{tot}	Φ^+	p_{Loss}^+/ρ^+
SCW1.0_90_20_001	1.56E+05	500	6.0	20.0	1.0	90°	-0.30	330	330	660	0.1216	1.2037
SCW1.0_90_20_002	1.56E+05	500	6.0	20.0	1.0	90°	-0.20	330	330	660	0.1004	1.2271
SCW1.0_90_20_003	1.56E+05	500	6.0	20.0	1.0	90°	-0.10	330	330	660	0.0801	1.2583
SCW1.0_90_20_004	1.56E+05	500	6.0	20.0	1.0	90°	0.00	330	330	660	0.0613	1.2984
SCW1.0_90_20_005	1.56E+05	500	6.0	20.0	1.0	90°	+0.05	330	330	660	0.0523	1.3215
SCW1.0_90_20_006	1.56E+05	500	6.0	20.0	1.0	90°	+0.10	330	330	660	0.0437	1.3463
SCW1.0_90_20_007	1.56E+05	500	6.0	20.0	1.0	90°	+0.15	330	330	660	0.0354	1.3724
SCW1.0_90_20_008	1.56E+05	500	6.0	20.0	1.0	90°	+0.20	330	330	660	0.0274	1.3996
SCW1.0_90_20_009	1.56E+05	500	6.0	20.0	1.0	90°	+0.25	330	330	660	0.0196	1.4277
SCW1.0_90_20_010	1.56E+05	500	6.0	20.0	1.0	90°	+0.30	330	330	660	0.0121	1.4565
SCW1.0_90_20_011	1.56E+05	500	6.0	20.0	1.0	90°	+0.35	330	330	660	0.0049	1.4858
SCW1.0_90_20_012	1.56E+05	500	6.0	20.0	1.0	90°	+0.40	330	330	660	0.0000	1.5134
SCW1.0_90_40_007	1.96E+05	500	6.0	40.0	1.0	90°	+0.15	350	310	660	0.0355	2.4994
SCW1.0_90_40_008	1.96E+05	500	6.0	40.0	1.0	90°	+0.20	350	310	660	0.0274	2.5266
SCW1.0_90_40_009	1.96E+05	500	6.0	40.0	1.0	90°	+0.25	350	106	456	0.0197	2.5547
SCW1.0_90_40_010	1.96E+05	500	6.0	40.0	1.0	90°	+0.30	350	286	636	0.0122	2.5835
SCW1.0_90_40_011	1.96E+05	500	6.0	40.0	1.0	90°	+0.35	350	310	660	0.0049	2.6128
SCW0.5_90_20_001	1.28E+05	500	6.0	20.5	0.5	90°	-0.20	242	242	484	0.0556	1.1319
SCW0.5_90_20_002	1.28E+05	500	6.0	20.5	0.5	90°	-0.10	242	242	484	0.0457	1.1345
SCW0.5_90_20_003	1.28E+05	500	6.0	20.5	0.5	90°	0.00	242	242	484	0.0358	1.1375
SCW0.5_90_20_004	1.28E+05	500	6.0	20.5	0.5	90°	+0.05	242	242	484	0.0308	1.1391
SCW0.5_90_20_005	1.28E+05	500	6.0	20.5	0.5	90°	+0.10	242	242	484	0.0258	1.1408
SCW0.5_90_20_006	1.28E+05	500	6.0	20.5	0.5	90°	+0.15	242	242	484	0.0209	1.1426
SCW0.5_90_20_007	1.28E+05	500	6.0	20.5	0.5	90°	+0.20	242	242	484	0.0159	1.1445
SCW0.5_90_20_008	1.28E+05	500	6.0	20.5	0.5	90°	+0.25	242	242	484	0.0110	1.1464
SCW0.5_90_20_009	1.28E+05	500	6.0	20.5	0.5	90°	+0.30	242	242	484	0.0061	1.1484
SCW0.5_90_40_006	1.68E+05	500	6.0	40.5	0.5	90°	+0.15	350	310	660	0.0209	2.5532
SCW0.5_90_40_007	1.68E+05	500	6.0	40.5	0.5	90°	+0.20	350	106	456	0.0160	2.5719
SCW0.5_90_40_008	1.68E+05	500	6.0	40.5	0.5	90°	+0.25	350	106	456	0.0110	2.5912
SCW1.0_45_20_001	1.56E+05	500	6.0	20.0	1.0	45°	0.00	330	330	660	0.0513	1.3261
SCW1.0_45_20_002	1.56E+05	500	6.0	20.0	1.0	45°	+0.10	330	330	660	0.0372	1.3671
SCW1.0_45_20_003	1.56E+05	500	6.0	20.0	1.0	45°	+0.15	330	330	660	0.0302	1.3899
SCW1.0_45_20_004	1.56E+05	500	6.0	20.0	1.0	45°	+0.20	330	330	660	0.0234	1.4140
SCW1.0_45_20_005	1.56E+05	500	6.0	20.0	1.0	45°	+0.25	330	330	660	0.0166	1.4392
SCW1.0_45_20_006	1.56E+05	500	6.0	20.0	1.0	45°	+0.30	330	330	660	0.0100	1.4653

Table 15: CFD Cases Simulation Data Part 2.

Case	#cells	*Re*	L_{outlet}^{+}	L_{intlet}^{+}	SCW^{+}	α	Δp^{+}	$t^{+}_{ave,start}$	t^{+}_{ave}	t^{+}_{tot}	Φ^{+}	p_{Loss}^{+}/ρ^{+}
SCW1.0_45_40_001	1.96E+05	500	6.0	40.0	1.0	45°	0.00	350	310	660	0.0513	2.4531
SCW1.0_45_40_002	1.96E+05	500	6.0	40.0	1.0	45°	+0.10	350	310	660	0.0372	2.4940
SCW1.0_45_40_003	1.96E+05	500	6.0	40.0	1.0	45°	+0.15	350	310	660	0.0303	2.5168
SCW1.0_45_40_004	1.96E+05	500	6.0	40.0	1.0	45°	+0.20	350	310	660	0.0234	2.5409
SCW1.0_45_40_005	1.96E+05	500	6.0	40.0	1.0	45°	+0.25	350	310	660	0.0166	2.5662
SCW1.0_45_40_006	1.96E+05	500	6.0	40.0	1.0	45°	+0.30	350	310	660	0.0100	2.5922
SCW0.5_45_20_001	1.28E+05	500	6.0	20.5	0.5	45°	0.00	330	330	660	0.0242	1.4142
SCW0.5_45_20_002	1.28E+05	500	6.0	20.5	0.5	45°	+0.10	330	330	660	0.0175	1.4389
SCW0.5_45_20_003	1.28E+05	500	6.0	20.5	0.5	45°	+0.15	330	330	660	0.0141	1.4520
SCW0.5_45_20_004	1.28E+05	500	6.0	20.5	0.5	45°	+0.20	330	330	660	0.0107	1.4655
SCW0.5_45_20_005	1.28E+05	500	6.0	20.5	0.5	45°	+0.25	330	330	660	0.0073	1.4794
SCW0.5_45_20_006	1.28E+05	500	6.0	20.5	0.5	45°	+0.30	330	330	660	0.0039	1.4936
SCW0.5_45_40_001	1.68E+05	500	6.0	40.5	0.5	45°	0.00	350	310	660	0.0242	2.5412
SCW0.5_45_40_002	1.68E+05	500	6.0	40.5	0.5	45°	+0.10	350	310	660	0.0175	2.5659
SCW0.5_45_40_003	1.68E+05	500	6.0	40.5	0.5	45°	+0.15	350	310	660	0.0141	2.5790
SCW0.5_45_40_004	1.68E+05	500	6.0	40.5	0.5	45°	+0.20	350	310	660	0.0107	2.5926
SCW0.5_45_40_005	1.68E+05	500	6.0	40.5	0.5	45°	+0.25	350	310	660	0.0073	2.6064
SCW0.5_45_40_006	1.68E+05	500	6.0	40.5	0.5	45°	+0.30	350	310	660	0.0039	2.6206

A.6.2 Laminar Inlet Profile

For the generation of laminar inlet profiles following utility settings are used:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      setInletVelocityDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

patch  inC; // specifies the name of the inlet patch
center  (-2.5 0.0 0.0); // specifies the center of the inlet patch

N       20;        // number of terms when approximating infinite sum (I am sure 20
is unnecesarily high but...)
u_av    1.0;       // average inlet velocity
heightDir (0 1 0); // specifies the direction in which 'height' is measured
widthDir  (0 0 1); // specifies the direction in which 'width' is measured. must be
perpendicular to heightDir
height    1.0;       // total length of inlet channel along the axis defined by the
exit flow (i.e.. x. I have been calculating Re based on this dimension)
width     1.0;       // total length of inlet channel along the axis perpendicular
to the exit flow (i.e.. z. for large aspect ratios. this gets bigger)
radius    0.5;
isRectangular;        //set if you want to set for a rectangular geometry. if not
set. patch is assumed to be circular
// ************************************************************************* //
```

A.6.3 Mapped Inlet Profile

To realize a mapped inlet condition following OpenFOAM® input needs to be provided:

0/U

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);
```

```
boundaryField
{
    inC
    {
        type                mapped;
        value               uniform (1 0 0);
        interpolationScheme cell;
        setAverage          true;
        average             (1 0 0);
    }

    outC
    {
        type            inletOutlet;
        inletValue      uniform (0 0 0);
        value           uniform (0 0 0);
    }
    outJ
    {
        type            inletOutlet;
        inletValue      uniform (0 0 0);
        value           uniform (0 0 0);
    }
    wallPatch
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
}
// ************************************************************************* //
```

constant/blockMeshDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

//Channel heigh and width
lwp 0.5; //Halth width. +x2
lwm -0.5; //Halth width. -x2
lhp  0.5; //Halth height. +x3
lhm -0.5; //Halth height. -x3
nw 20; //Number cells width. x2
nh 20; //Number cell height. x3

//Channel main segments: a. b. c
lai -2.5; //negative x1 position of segment a
lci -0.5; //negative x1 position of segment c
lco 0.5; //positive x1 position of segment c
lbo 6.5; //positive x1 position of segment b
na 40; //Number of cells of a. x1
nc 20; //Number of cells of c. x1
nb 120; //Number of cells of b. x1

vertices
```

```
(
    ($lai $lwm $lhm)      //0
    ($lci $lwm $lhm)      //1
    ($lci $lwp $lhm)      //2
    ($lai $lwp $lhm)      //3

    ($lai $lwm $lhp)      //4
    ($lci $lwm $lhp)      //5
    ($lci $lwp $lhp)      //6
    ($lai $lwp $lhp)      //7

    ($lco $lwm $lhm)      //8
    ($lbo $lwm $lhm)      //9
    ($lbo $lwp $lhm)      //10
    ($lco $lwp $lhm)      //11

    ($lco $lwm $lhp)      //12
    ($lbo $lwm $lhp)      //13
    ($lbo $lwp $lhp)      //14
    ($lco $lwp $lhp)      //15
);

blocks
(
    hex (0 1 2 3 4 5 6 7) ($na $nw $nh) simpleGrading (1 1 1)
    hex (1 8 11 2 5 12 15 6) ($nc $nw $nh) simpleGrading (1 1 1)
    hex (8 9 10 11 12 13 14 15) ($nb $nw $nh) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
    inC
    {
        type        mappedPatch; // see pitzDailyMapped Case
        offset          ( 6 0 0 );
        sampleRegion    region0;
        sampleMode      nearestCell;
        samplePatch     none;
        faces           ((0 4 7 3));
    }
    outC
    {
        type            patch;
        faces           ((9 10 14 13));
    }
    outJ
    {
        type            patch;
        faces           ((1 8 11 2));
    }

    sideWall
    {
      type          wall;
      faces         (
       (0 1 5 4)
       (1 8 12 5)
       (8 9 13 12)

       (3 2 6 7)
       (2 11 15 6)
       (11 10 14 15)
                    );
    }

    bottomWall
    {
      type          wall;
      faces         (
       (4 5 6 7)
       (5 12 15 6)
       (12 13 14 15)
       (0 1 2 3)
       (8 9 10 11)
                    );
    }
);

mergePatchPairs
(
);
// ************************************************************************* //
```

constant/boundary

The usage of the settings in the *blockMeshDict* results in following boundary conditions:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       polyBoundaryMesh;
    location    "constant/polyMesh";
    object      boundary;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

4
(
    inC
    {
        type            mappedPatch;
        inGroups        1(mappedPatch);
        nFaces          400;
        startFace       452000;
        sampleMode      nearestCell;
        sampleRegion    region0;
        samplePatch     none;
        offsetMode      uniform;
        offset          (6 0 0);
    }
    outC
    {
        type            patch;
        nFaces          400;
        startFace       452400;
    }
    outJ
    {
        type            patch;
        nFaces          400;
        startFace       452800;
    }
    wallPatch
    {
        type            wall;
        inGroups        1(wall);
        nFaces          30800;
        startFace       453200;
    }
)
// ************************************************************************* //
```

A.6.4 Comparison of Laminar and Mapped Inlet Profiles

Sample lines are taken during the simulations. Here the results of two sample lines, one at the main channel inlet along the y^+-axis ([-2.49 -0.5 0.0] – [-2.49 0.5 0.0]) and one at the main channel inlet along the z^+-axis ([-2.49 0.0 -0.5] – [-2.49 0.0 0.5]), are shown. The ±5% curves are calculated by adding ±5% to the laminar curve data. It can be seen that the results for the laminar inlet and the mapped inlet (*Re*=500) are very similar.

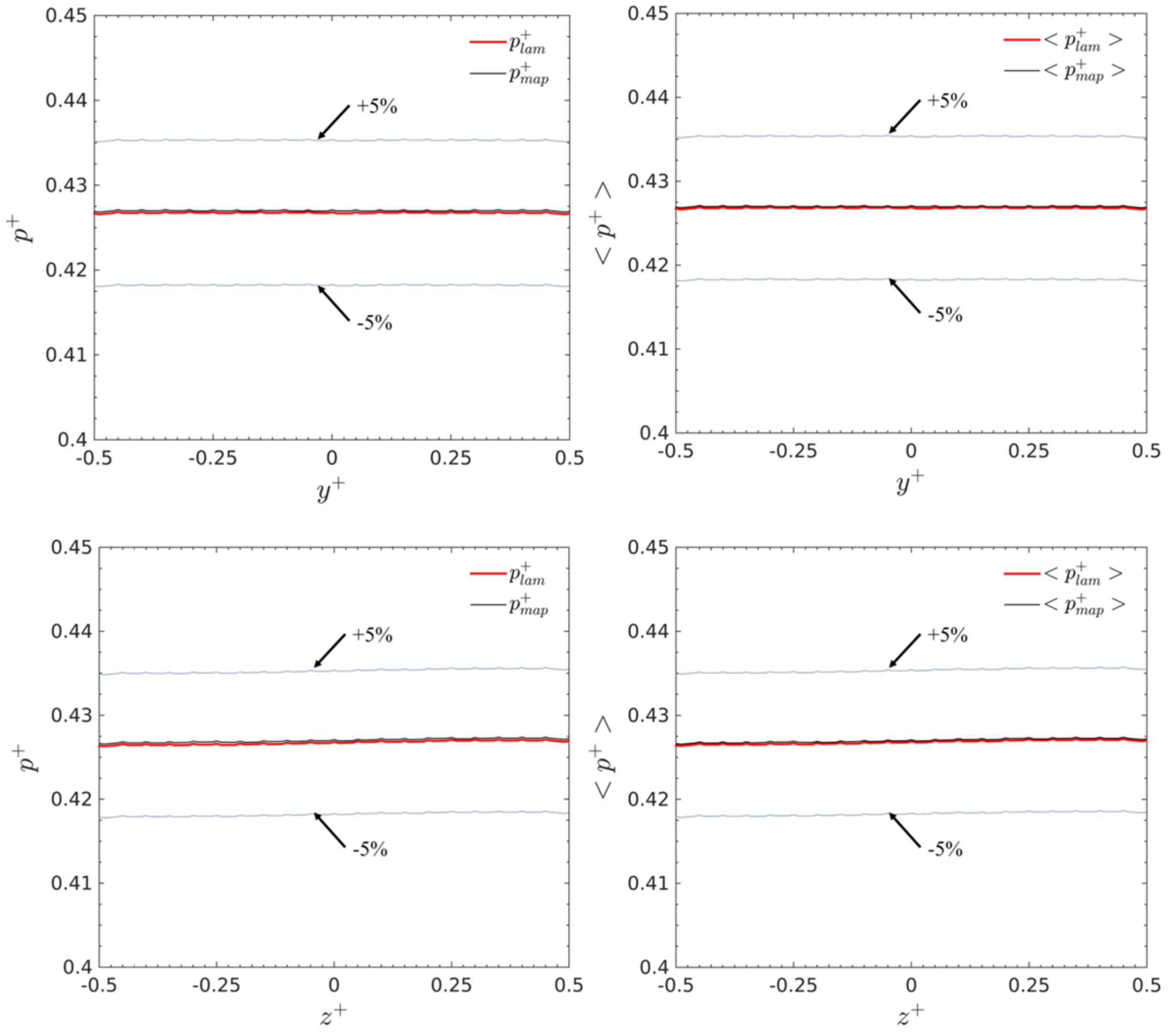


Fig. 46: Comparison of the pressure and the time-averaged pressure along a sample line. Top: sample line in y^+-direction. Bottom: sample line in z^+-direction (case: SCW1.0_90_20_10)

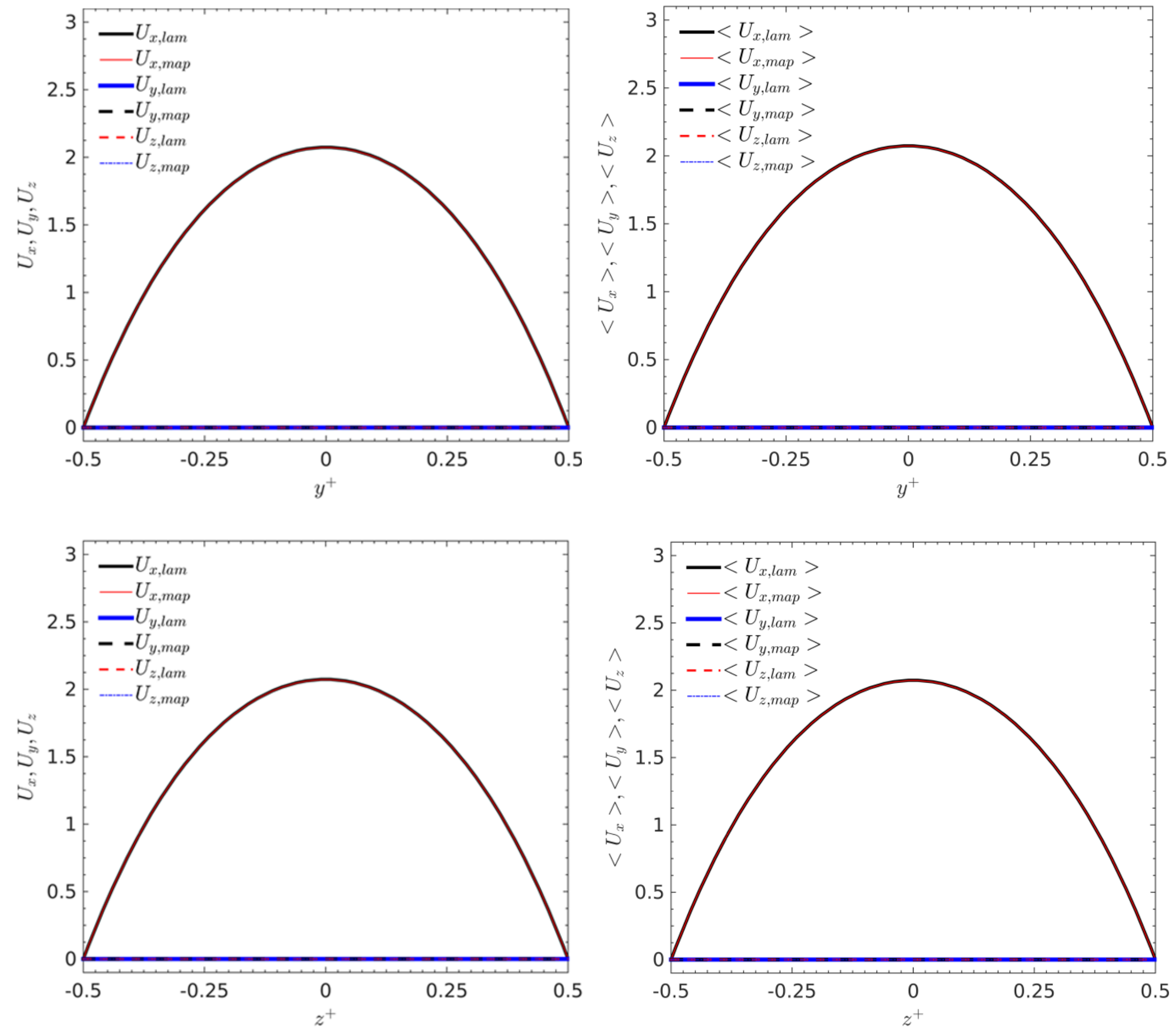


Fig. 47: Comparison of the velocities and the time-averaged velocities along a sample line. Top: sample line in y^+-direction. Bottom: sample line in z^+-direction (case: SCW1.0_90_20_10)

A.6.5 Steady State Case SCW1.0_90_20_10

The approach to steady state over ~28·10^3 iterations or 330 dimensionless time units is shown in Fig. 48 and Fig. 49.

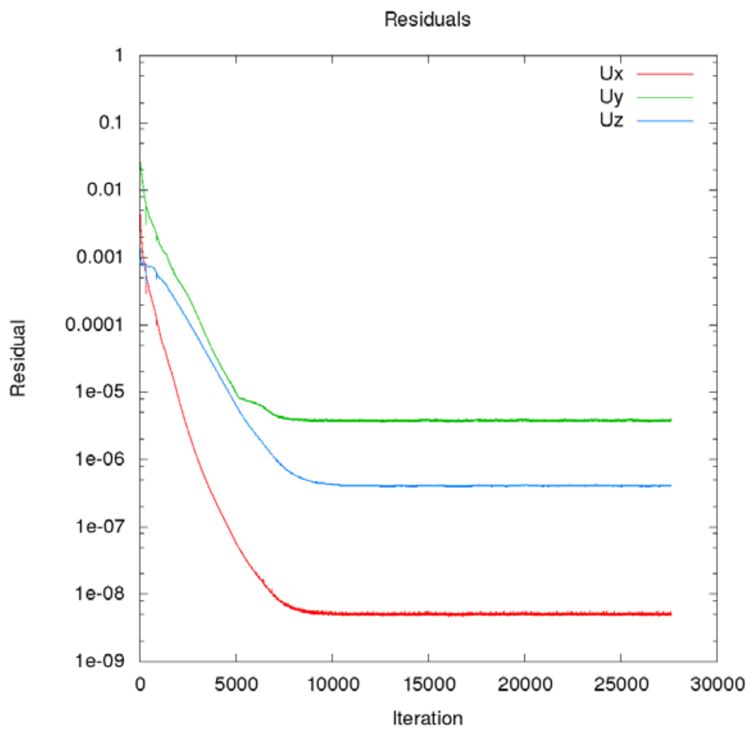


Fig. 48: Residuals over ~28·10^3 iterations.

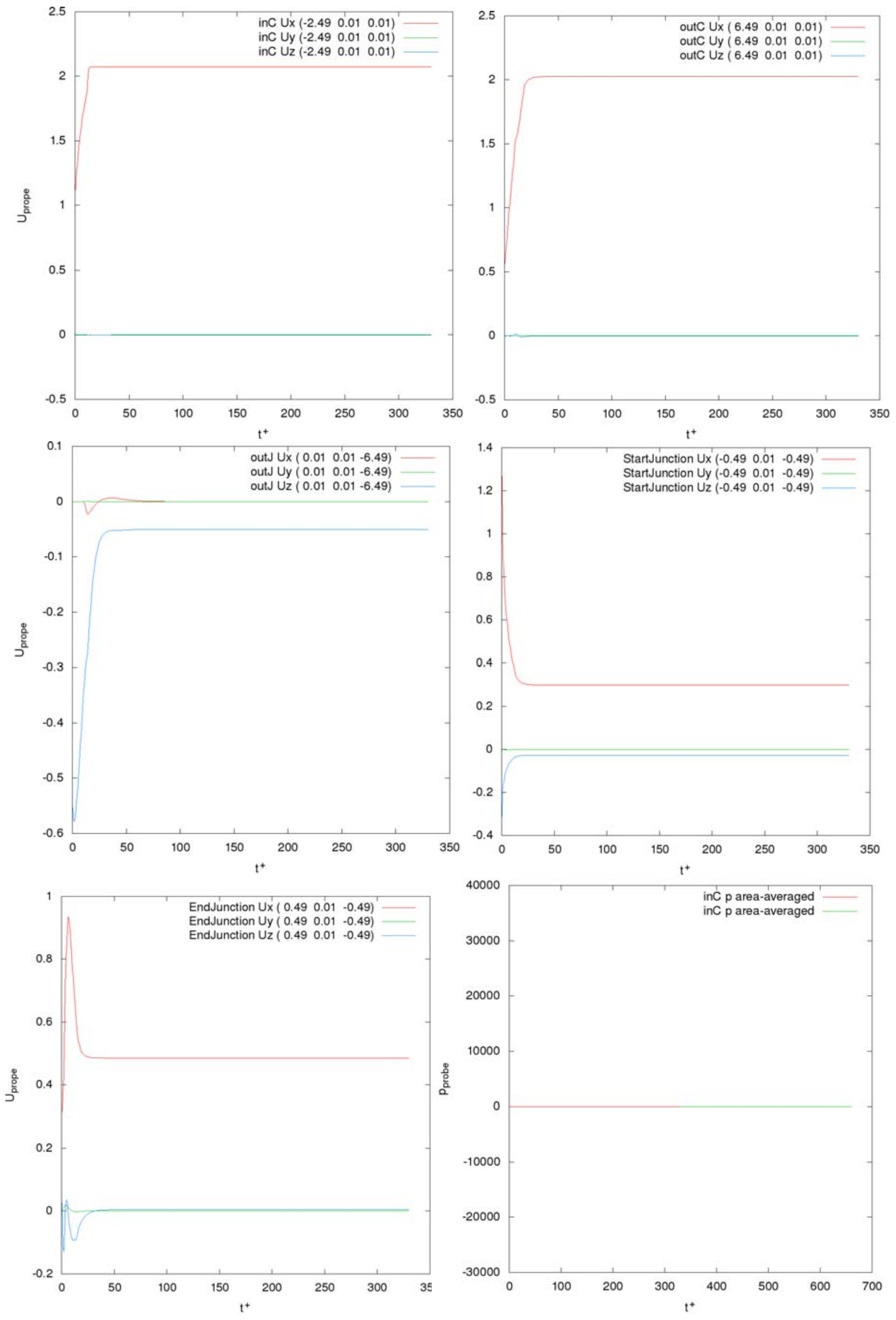


Fig. 49: Velocity and pressure at different probe positions in T-junction. 1st row: Velocity components in x^+-, y^+-, z^+-direction at the beginning and the end of the main channel. 2nd row: Velocity components in x^+-, y^+-, z^+- direction at the end of the side channel and right at the beginning of the junction. 3rd row: Velocity components in x^+-, y^+-, z^+-direction right at the end of the junction and the area-averaged pressure at the inlet of the main channel.

Appendix B Fibre Motion

The translational and rotational fibre motion can be described by using Newton's Equation of motion, which can be written as [29], [31]:

$$m\frac{\partial \mathbf{v}}{\partial t} = \mathbf{F}^{\mathbf{h}} + \mathbf{F}^{\mathbf{c}} \qquad \frac{\partial}{\partial t}\left(\mathbf{I}\cdot\boldsymbol{\omega}\right) = \mathbf{T}^{\mathbf{h}} + \mathbf{T}^{\mathbf{c}} \tag{B.1.1}$$

As explained in chapter 3.2.1 the fibres are assumed to be rigid spherocylinders for the contact force calculations, but for the hydrodynamic force model the fibres are assumed to be prolate spheroids. To now link those two geometries Cox's equation [36] for replacing a cylindrical fibre (length $d_{Major}=l$, diameter $d_{Minor}=d$) with a prolate spheroid (same length $d_{Major}=l$, half-length of the minor axis b), is used:

$$b = \frac{d\sqrt{\ln\frac{l_i}{d}}}{2.48} \tag{B.1.2}$$

Newton's Equation of motion can then be resolved with $m_i = 4/3\ \pi\ \rho\ d_{Major}\ d_{Minor}^2$ representing the mass of the fibre, **v** the velocity vector in its center of mass, t the time, **I** the tensor of inertia, $\mathbf{F}^{\mathbf{h}}$ and $\mathbf{T}^{\mathbf{h}}$ the force and the torque due to hydrodynamic forces, $\mathbf{F}^{\mathbf{c}}$ and $\mathbf{T}^{\mathbf{c}}$ the force and the torque resulting from collisions [29], [31]. Body forces, e.g. gravity, are neglected.

B.1 Eccentricity of Prolate Spheroids

The deviation from the prolate spheroid from a sphere is described by the eccentricity [55]. Lindström [31] describes the eccentricity of a prolate spheroid by:

$$e = \sqrt{1 - \frac{b^2}{a^2}} \tag{B.1.3}$$

The eccentricity parameters needed for the hydrodynamic force and torque calculations as:

$$L(e) = \ln\left(\frac{1+e}{1-e}\right) \tag{B.1.4}$$

$$X^A\left(e\right) = \frac{8}{3}e^3\left[-2e + (1+e^2)L(e)\right]^{-1} \tag{B.1.5}$$

$$X^C\left(e\right) = \frac{4}{3}e^3\left(1-e^2\right)\left[2e - (1-e^2)L(e)\right]^{-1} \tag{B.1.6}$$

$$Y^A\left(e\right)=\frac{16}{3}e^3\left[2e+(3e^2-1)L(e)\right]^{-1} \tag{B.1.7}$$

$$Y^C\left(e\right)=\frac{4}{3}e^3\left(2-e^2\right)\left[-2e+(1+e^2)L(e)\right]^{-1} \tag{B.1.8}$$

$$Y^H\left(e\right)=\frac{4}{3}e^5\left[-2e+(1+e^2)L(e)\right]^{-1} \tag{B.1.9}$$

B.2 Hydrodynamic Force F^h *and Torque* T^h

If the particle Reynolds number Re_p is small, the hydrodynamic drag forces are dominated by viscous effects and the inertial effects can be neglected ($\mathbf{F^h} = \mathbf{F^v}$).

B.2.1 Viscous Effects

Lindström [31] describes the force and the torque as follows:

$$\mathbf{F}^v=\mathbf{A}^v\cdot[\mathbf{u}-\mathbf{v}] \tag{B.2.1}$$

$$\mathbf{T}^v=\mathbf{C}^v\cdot[\mathbf{\Omega}-\boldsymbol{\omega}\ ]+\mathbf{H}^v\cdot\dot{\boldsymbol{\gamma}} \tag{B.2.2}$$

A^v, C^v, H^v are hydrodynamic resistance tensors for the viscous effects:

$$\begin{aligned}\mathbf{A}^v&=3\pi\eta l[Y^A\boldsymbol{\delta}+\left(X^A-Y^A\right)x''x'']\\ \mathbf{C}^v&=\pi\eta l^3[Y^C\boldsymbol{\delta}+\left(X^C-Y^C\right)x''x'']\\ \mathbf{H}^v&=-\pi\eta l^3Y^H(\boldsymbol{\varepsilon}\cdot x'')x''x''\end{aligned} \tag{B.2.3}$$

$\dot{\gamma}$ is the rate of the strain tensor $\dot{\gamma}$ =[∇u+(∇u^T)]/2, u and v are the fluid and fibre velocity and Ω and ω are the angular velocity of the fluid and the fibre, all at the center of mass of the prolate spheroid, δ and ε are the unit tensor and the permutation tensor.

B.3 Contact Forces F^c and Torque T^c

An established model to determine the contact force $\mathbf{F^c}$ on particles, in granular systems in DEM, is the linear spring-dashpot model in normal and tangential direction of a contact [26], [39]. The general contact force in a spring-dashpot model for two interacting particles i and j is given by:

$$\mathbf{F}^{c}=\underbrace{\left(k_n \underbrace{\delta\mathbf{n_{ij}}}_{\substack{\text{normal}\\ \text{overlap}}}-\gamma_n \underbrace{\mathbf{vn_{ij}}}_{\substack{\text{normal rel.}\\ \text{velocity}}}\right)}_{\text{normal force }\mathbf{F_n}}+\underbrace{\left(k_t \underbrace{\delta\mathbf{t_{ij}}}_{\substack{\text{tangential}\\ \text{overlap}}}-\gamma_t \underbrace{\mathbf{v\,t_{ij}}}_{\substack{\text{tangetial}\\ \text{rel. velocity}}}\right)}_{\text{tangential force }\mathbf{F_t}\le x_\mu \mathbf{F_n}} \tag{B.3.1}$$

The force consists of a normal $\mathbf{F_n}$ and a tangential $\mathbf{F_t}$ force. Both can be decomposed in a linear repulsive and a linear dissipative force, with a spring stiffness in normal and tangential direction k_n and k_t, a viscous damping coefficient in normal and tangential direction γ_n and γ_t, a relative velocity in normal $\boldsymbol{vn_{ij}}$ and tangential $\boldsymbol{vt_{ij}}$ direction, and the overlap in normal $\delta\boldsymbol{n_{ij}}$ and tangential $\delta\boldsymbol{t_{ij}}$ direction [26], [56].

In our model which is based on LIGGGHTS® Hooke/Stiffness-model [56], two regions have to be distinguished for the linear-spring-dashpot model, the roughness layer region surrounding each fibre (see chapter 3.2.2) and the fibre region in which an actually overlap of the fibres occurs. The considered forces in our model are illustrated in Fig. 50. The basis of the model are taken from Lindström [31].

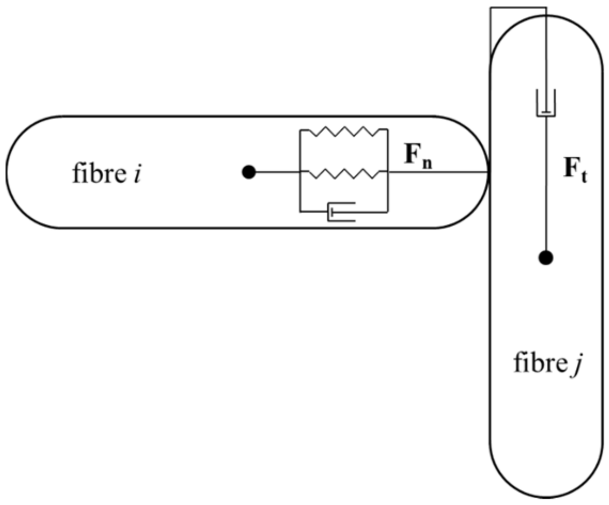


Fig. 50: Schematic representation of linear spring-dashpot model used for fibre interaction. Graphic design taken from Goniva [57].

B.3.1 Contact Forces in the Roughness Layer Region

In this region there is only a spring force in normal direction used $\mathbf{F_{n.s.roughness}}$, there is no tangential force or damping in normal direction considered. The value of the spring constant is a factor ($k_n\, x_{stiffness}$) of the spring constant used for the fibre region. The normal spring force in the roughness layer is thus be calculated by:

$$\mathbf{F_{n,s,roughness}} = k_n\, x_{stiffness}\, \delta_{roughness} \mathbf{n_{ij}} \tag{B.3.2}$$

B.3.2 Contact Forces in the Fibre Contact Region

In this region the fibres are actually overlapping, they have passed the roughness layer completely. In our model there is a spring force and a damping (dashpot) force in normal direction, but in tangential direction there is only a damping (dashpot) force.

<u>Normal Force</u>

The total normal force is the sum of the normal spring and the normal tangential force:

$$\mathbf{F_n} = \mathbf{F_{n,s}} + \mathbf{F_{n,t}} \tag{B.3.3}$$

The normal spring force is the sum of the normal spring force from the roughness layer region and the normal spring force from the fibre region:

$$\mathbf{F_{n,s}} = \mathbf{F_{n,s,roughness}} + \mathbf{F_{n,s,fibre}} \tag{B.3.4}$$

The normal spring force in the fibre region is determined by:

$$\mathbf{F_{n,s,fibre}} = k_n \, \delta_{fibre} \mathbf{n_{ij}} \tag{B.3.5}$$

The damping force in normal direction is calculated by:

$$\mathbf{F_{n,d}} = \mathbf{F_{n,d,fibre}} = \gamma_t \mathbf{vt_{ij}} \tag{B.3.6}$$

<u>Tangential Force</u>

The tangential force consists only of a damping force/ frictional force, there is no spring force determined. The tangential damping force is limited by the possible frictional force, with x_μ being the friction coefficient.

$$\mathbf{F_t} = \mathbf{F_{t,d}} = \gamma_t \mathbf{vt_{ij}} \qquad \mathbf{F_{t,d}} \leq x_\mu \mathbf{F_n} \tag{B.3.7}$$

So now the total contact force is:

$$\mathbf{F^c} = \mathbf{F_n} + \mathbf{F_t} \tag{B.3.8}$$

Torque

The torque on the fibre due to the contact is determined by the cross product of the distance from the contact point to the center of mass and the force acting on the contact point:

$$\mathbf{T}^c = \mathbf{r} \times \mathbf{F}^c \tag{B.3.9}$$

Note, this is used for fibre-fibre and fibre-wall interaction.

B.4 Lubrication Force

The lubrication force is based on Lindström's lubrication force model [31]. In this model the lubrication force of two interacting cylinders *i* and *j* is described (see Fig. 51). This model can also be used for cylinder wall collisions. The lubrication force is limited to avoid instability in the simulation if two particles' distance and angle is close to zero, thus two cases are distinguished.

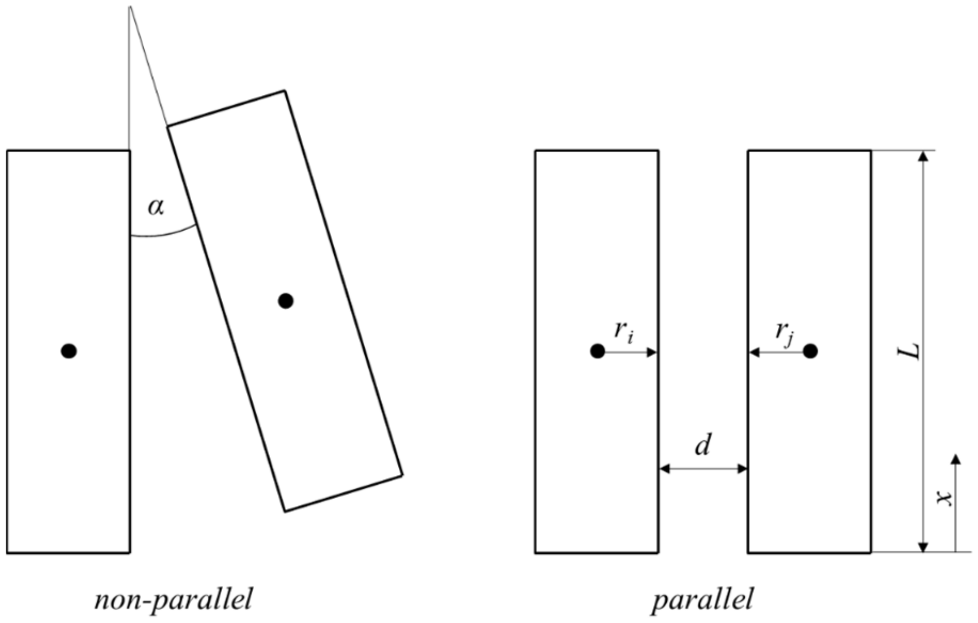


Fig. 51: Two approaching cylinders. Left panel: cylinders are non-parallel, angle α. Right panel: cylinders are parallel.

The effective radius used to calculate the lubrication force can be determined by:

$$R_{eff} = \frac{2}{\frac{1}{r_i} + \frac{1}{r_j}} \tag{B.4.1}$$

with r_i and r_j being the radius of the two cylinders. Under consideration of the limiter the lubrication force can thus be determined as:

$$F_{\text{lub},\alpha} = -\eta \frac{12\pi}{\sin\alpha} \frac{{R_{eff}}^2}{d} \tag{B.4.2}$$

$$\frac{d}{dx} F_{\text{lub},||} = -\eta \left(A_0 + A_1 \frac{d}{R_{eff}} \right) \left(\frac{d}{R_{eff}} \right)^{-\frac{3}{2}} \qquad A_0 = 3\pi\sqrt{2}/8 \quad \mathrm{A}_1 = 207\pi\sqrt{2}/160 \tag{B.4.3}$$

$$F_{\text{lub}} = \begin{cases} F_{\text{lub},\alpha} & \mathit{if} \left|F_{\text{lub},\alpha}\right| \le \left|L\frac{d}{dx}F_{\text{lub},\|}\right| \\ L\frac{d}{dx}F_{\text{lub},\|} & \mathit{if} \left|F_{\text{lub},\alpha}\right| \ge \left|L\frac{d}{dx}F_{\text{lub},\|}\right| \end{cases} \tag{B.4.4}$$

$$\mathbf{F_{lub}} = F_{\text{lub}}\dot{\mathbf{d}} \tag{B.4.5}$$

Here d is the distance between the closest points, α the angle between the cylinders, η the viscosity of the fluid, L the infinite length of the cylinders.

B.5 Contact Point Detection

In Appendix B the forces and torques due to fibre-fibre and fibre-wall interactions are described. In this chapter it is shown how the closest distance between two fibres or a fibre and a wall and therefore a possible contact is determined. The code used for the computation of the distances and contacts is based on Schneider and Eberly [30].

B.5.1 Line-Line Distance

To determine the distance between two fibres and their contact, the fibres can be simplified with the spherosimplicies method [28] as line segments [29]. Line segments can be described by a basis point $\mathbf{P_i}$, a direction $\mathbf{d_i}$ and a length s (or t), e.g. two line segments are: $L_0(s) = \mathbf{P_0} + s\mathbf{d_0}$ and $L_1(t) = \mathbf{P_1} + t\,\mathbf{d_1}$. The two points, one on each line segment, with the smallest distance are $\mathbf{Q_0}$ and $\mathbf{Q_1}$. The length of the vector $\mathbf{v=Q_0-Q_1}$ represents the smallest distance (see Fig. 52).

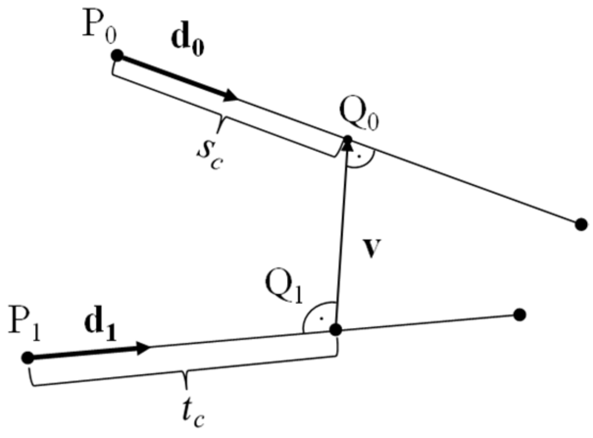


Fig. 52: Distance between two line segments [30].

The first step to find the distance and a possible contact point is to determine whether the two line segments are parallel or not. This can be done by calculating the dot product of the two orientation vectors $\mathbf{d_0}$ and $\mathbf{d_1}$. If $\mathbf{d_0}\cdot\mathbf{d_1}=0$ then the two lines are parallel, if $\mathbf{d_0}\cdot\mathbf{d_1}\neq 0$ then they are non-parallel. In both cases the minimum distance is determined by minimizing the square distance function Q(s,t):

$$Q(s,t) = \left\| L_o(\mathrm{s}) - \mathrm{L}_1(t) \right\|^2 = as^2 + 2bst + ct^2 + 2ds + 2et + f \qquad \text{(B.5.1)}$$

with a = $\mathbf{d_0}\cdot\mathbf{d_0}$, b = $-\mathbf{d_0}\cdot\mathbf{d_1}$, c = $\mathbf{d_1}\cdot\mathbf{d_1}$, d = $\mathbf{d_0}\cdot$(P0-P1), e = $\mathbf{d_1}\cdot(P_0-P_1)$, f =$(P_0-P_1)\cdot(P_0-P_1)$.

Although depending on whether the two lines are parallel or not, the function is either a parabolic cylinder ($ac-b^2=0$) or a paraboloid ($ac-b^2>0$). Also, it needs to be considered that mathematically a line segment is a specific part of a line, which restricts the domain for *s* and *t* to [0,1] (see Fig. 53 left panel), hence the square distance function needs to be minimized over a unit square $[0,1]^2$ [30].

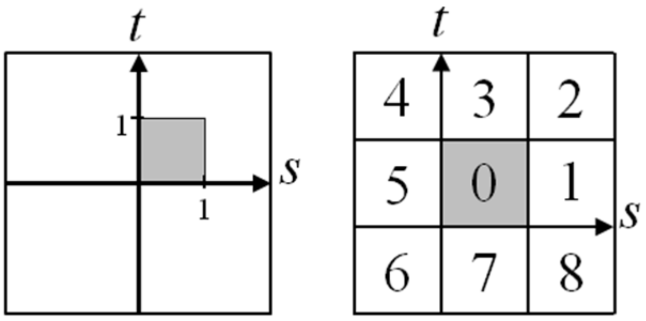


Fig. 53: Left panel: domain for *s* and *t*. Right panel: Domain region 0 for s and t and boundary domains region1-8 [30].

Non-Parallel Line Segments

In case of non-parallel line segments the minimum of Q is at $s_c=(be-cd)/(ac-b^2)$ and $t_c=(bd-ae)/(ac-b^2)$. The minimum distance is thus be found if s_c and t_c are within their domain [0,1] (region 0), which represents points with minimum distance within the line segments. Otherwise s_c and t_c are at one of the boundaries of the square domain (see Fig. 53 right panel), here the region 1-8 need to be distinguished. Regions 1-8 represent the situation that either both or one of the closest points is at the end point of a line segment.

Parallel Line Segments

The two line segments lie on parallel lines, so that line segment L_1 can be projected on L_0 to describe the end point of the second line segment as $\mathbf{P_1}=\mathbf{P_0}+\sigma_0\mathbf{d_0}+\mathbf{u_0}$, with $\mathbf{u_0}$ being orthogonal to $\mathbf{d_0}$ and $\sigma_0=-\ d/a$. Further $\mathbf{P_1}+\mathbf{d_1}=\mathbf{P_0}+\sigma_1\mathbf{d_0}+\mathbf{u_1}$, with $\mathbf{u_1}$ being orthogonal to $\mathbf{d_0}$ and $\sigma_1=-(\ b+d)/a$. If the relative position [min(σ_0, σ_1), max(σ_0, σ_1)] is now within [0,1] then there are multiple possible points for the minimum distance, otherwise the minimum distance is located at the end points.

In both cases, parallel and non-parallel, the points with the closest distance within the line segment (or at the end of the line segments) can thus be found. If two fibres *i* and *j* approach each other and the distance between the closest points becomes less than $d_{Minor,i}/2+l_{roughness,i}$ + $d_{Minor,j}/2+l_{roughness,j}$ a contact occurs.

B.5.2 Line-Triangle Distance

Similar to the line-line distance and contact determination, the fibre-wall determination also uses the method of spherosimplicies [28]. Herby the fibre is again described as a line segment $L(t)=\mathbf{P}_0+t\mathbf{d}$, while the wall is represented by a triangle [29] $T(u,v)=\mathbf{V}_0+u\mathbf{e}_0+v\mathbf{e}_1$ with $\mathbf{e}_0=\mathbf{V}_1-\mathbf{V}_0$ and $\mathbf{e}_1=\mathbf{V}_2-\mathbf{V}_0$, where $\mathbf{V}_i$ are the vertices of the triangle (see Fig. 54). All points within the triangle can then be represented by *u* and *v* with $0 \leq u,\ v \leq 1$ and $u + v \leq 1$[30].

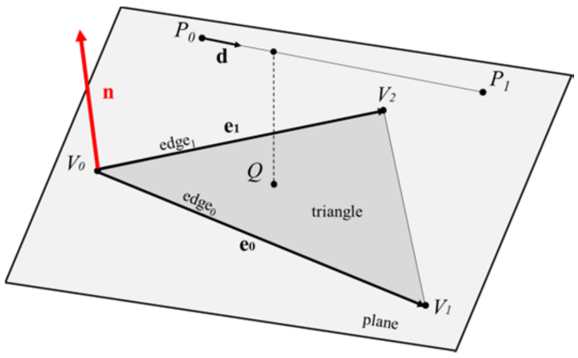


Fig. 54: Line-triangle distance determination [30]**.**

The first step is again to check whether the line and the triangle are parallel. This can be done by calculating the normal vector on the triangle and its plane $\mathbf{n}=\mathbf{e}_0\mathrm{x}\mathbf{e}_1$ and generating the dot product between the normal vector **n** and the direction vector of the line segment **d**. If the dot product is greater than zero, an interaction between the line and the plane might occur, but not necessarily of the line and the triangle, which is just part of the plane [30], [58].

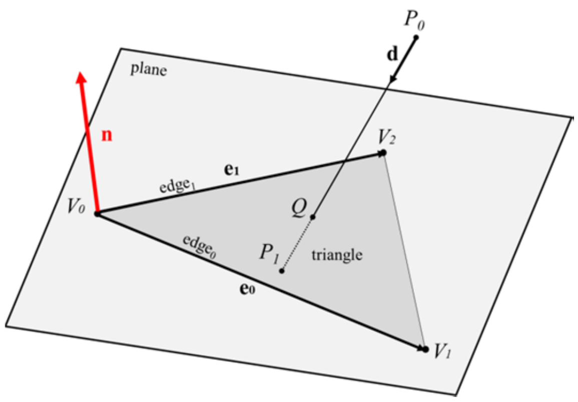


Fig. 55: Line-triangle contact determination [30], [58]**.**

The smallest distance is again determined by a minimization of the squared distance function:

$$\mathbf{Q(u,v,t)} = \left\|\mathbf{T(u,v) - L(t)}\right\|^2_{\min} \tag{B.5.2}$$

Compactly written as:

$$\begin{aligned}\mathbf{Q(u,v,t)} = \; & a_{00}\,u^2 + a_{11}\,v^2 + a_{22}\,t^2 \\ & + 2\,a_{01}\,u\,v + 2\,a_{02}\,u\,t + 2\,a_{12}\,v\,t \\ & + 2\,b_0\,u + 2\,b_1\,v + 2\,b_2\,t \\ & + c\end{aligned} \tag{B.5.3}$$

where:

$$\begin{array}{lll} a_{00} = \mathbf{e_0} \cdot \mathbf{e_0} & a_{11} = \mathbf{e_1} \cdot \mathbf{e_1} & a_{22} = \mathbf{d} \cdot \mathbf{d} \\ a_{01} = \mathbf{e_0} \cdot \mathbf{e_1} & a_{02} = -\mathbf{e_0} \cdot \mathbf{d} & a_{12} = -\mathbf{e_1} \cdot \mathbf{d} \\ b_0 = \mathbf{e_0} \cdot (\mathbf{V} - \mathbf{P}) & b_1 = \mathbf{e_1} \cdot (\mathbf{V} - \mathbf{P}) & b_2 = -\mathbf{d} \cdot (\mathbf{V} - \mathbf{P}) \\ c = (\mathbf{V} - \mathbf{P}) \cdot (\mathbf{V} - \mathbf{P}) & & \end{array} \tag{B.5.4}$$

Similar to the line-line approach the solution for *u, v, t* can mathematically be expressed in a three-dimensional result domain (see Fig. 56). In region 0 the closest points are within the line segment and within the triangle. In region 1-6 the line segment does not intersect the triangle, this can either be because the line segment and the triangle are parallel or the interaction point is on the plane outside the triangle. In both cases the closest point is on one of the three edges of the triangle, thus the minimum distance from all three edges to the line segment needs to be determined and the smallest gives then the closest points [30].

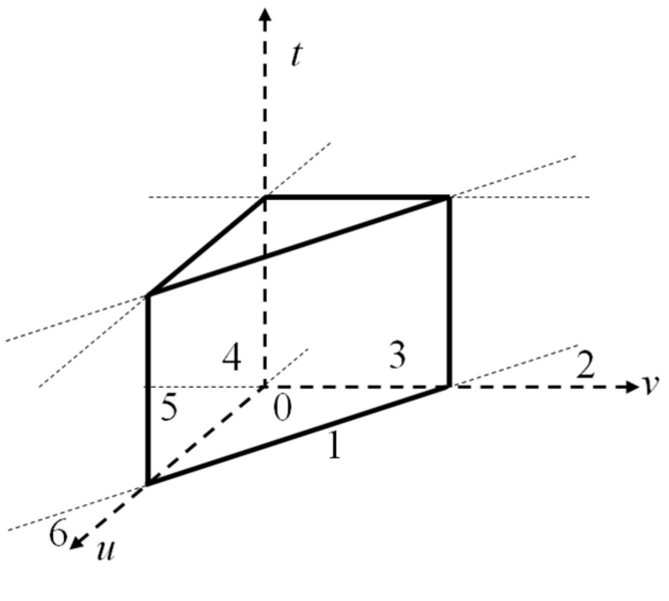


Fig. 56: Three dimensional solution space for line-triangle distance determination [30].

If the two closest points on the line segment and the triangle are determined and the distance between those two points is less than $d_{Minor}/2+l_{roughness}$ a contact occurs [29].

Appendix C DEM Simulation

C.1 Fibre-Fibre Interaction Test Case

To verify the fibre-fibre interaction LIGGGHTS® code (i.e., the numerical solution), implemented by S. Radl [29], an analytical solution for fibre-fibre collisions is used.

In the 2D analytical solution two identical fibres, referred to as fibre A and fibre B, collide in the x^+z^+-plane (rotation only around the y^+-axis). Fibre A has its center of mass $\mathbf{cm_A}$ at the origin of the world coordinate system (x^+, y^+, z^+). The center of mass of fibre B $\mathbf{cm_B}$ differs due to the angle α between the two fibres. Both fibres have initial (pre-collision) velocities $\mathbf{v_{A1}}$, $\mathbf{v_{B1}}$ and initial (pre-collision) angular velocities $\boldsymbol{\omega_{A1}}$, $\boldsymbol{\omega_{B1}}$.

Generally the analytical solution is divided into two sections. First the collision point $\mathbf{P_\alpha}$ as a function of the angle α between the two center lines of the fibres, the normal unit vector $\mathbf{e_{n.P\alpha}}$ at the collision point and the vectors from the center of mass of each fibre to the collision point are determined by using vector analysis [59]. Then the final (post-collision) velocities $\mathbf{v_{A2}}$, $\mathbf{v_{B2}}$ and the final (post-collision) angular velocities $\boldsymbol{\omega_{A2}}$, $\boldsymbol{\omega_{B2}}$ are calculated, according to an oblique, eccentric collision [60].

C.1.1 Collision Point and Contact Vector

To detect the fibres' collision point $\mathbf{P_\alpha}$, the fibres are considered to be spherocylinders. The starting point for the derivation of the collision point is an initial fibre position with the fibres' center lines being normal to each other ($\alpha = 0$). The collision point on fibre A is always on the lower half of the right half-sphere ending, while on fibre B it's always on the left side of the middle (cylindrical) body part. The position of $\mathbf{P_\alpha}$ on fibre B can be changed by variegating x (see Fig. 57).

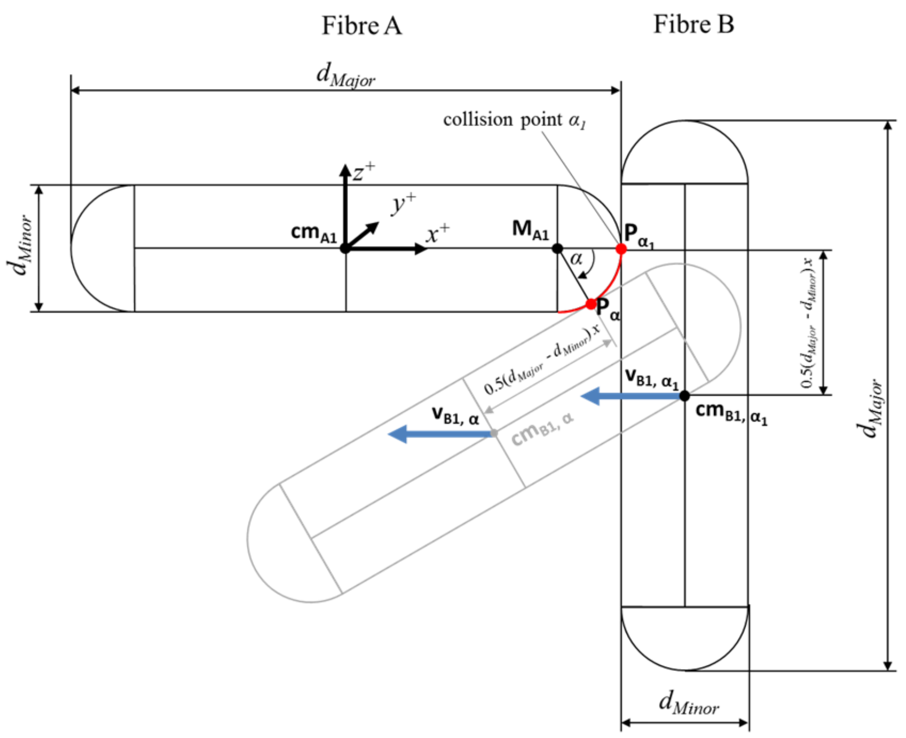


Fig. 57: Fibre-fibre collision areas. Red line shows the collision area of fibre A (does change due to angle *α*). Red dot shows collision point on fibre B (does not change due to the angle *α*, but due to *x*).

The first collision point ($\alpha = 0$) is located at $\mathbf{P}_{\alpha=0}$ [0.5 d_{Major} 0.0]. The rotation point of fibre B is equal to the center of fibre A's right half sphere center point $\mathbf{M}_{A1}$[0.5 ($d_{Major} - d_{Minor}$) 0.0]. The rotation angle is $0 \le \alpha < \pi/4$.

Calculation of vectors for *α = 0*

center of mass of fibre A (valid for all α): $\mathbf{cm}_{A1} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$

center of mass of fibre B, with x being a factor to vary the collision point on the left side of the cylindrical body of fibre B: $\mathbf{cm}_{B1} = \begin{bmatrix} 0.5\left(d_{Major} + d_{Minor}\right) & 0 & 0.5\left(d_{Major} - d_{Minor}\right)x \end{bmatrix}$

center of fibre A's right half sphere to collision point $\mathbf{P}_{\alpha=0}$: $\overline{\mathbf{M}_{A1}\mathbf{P}_{\alpha=0}} = \mathbf{P}_{\alpha=0} - \mathbf{M}_{A1}$

center of mass of fibre A to collision point $\mathbf{P}_{\alpha=0}$: $\overline{\mathbf{cm}_{A1}\mathbf{P}_{\alpha=0}} = \mathbf{M}_{A1} + \overline{\mathbf{M}_{A1}\mathbf{P}_{\alpha=0}}$ (C.1.1)

center of mass of fibre B to collision point $\mathbf{P}_{\alpha=0}$: $\overline{\mathbf{cm}_{B1}\mathbf{P}_{\alpha=0}} = \overline{\mathbf{cm}_{A1}\mathbf{P}_{\alpha=0}} - \mathbf{cm}_{B1}$

center of fibre A' right half sphere to center of mass of fibre B: $\overline{\mathbf{M}_{A1}\mathbf{cm}_{B1,\alpha=0}} = \mathbf{cm}_{B1} - \mathbf{M}_{A1}$

normal vector for collision – tangent on half sphere in $\mathbf{P}_{\alpha=0}$: $\mathbf{e}_{n,\mathbf{P}_{\alpha=0}} = \dfrac{\overline{\mathbf{M}_{A1}\mathbf{P}_{\alpha=0}}}{\left|\overline{\mathbf{M}_{A1}\mathbf{P}_{\alpha=0}}\right|}$

Calculation of vectors for $\alpha \neq 0$

The rotation of fibre B can be realized by using the rotation matrix [59] around the y^+-axis in $\mathbf{M_{A1}}$:

$$\mathbf{R_y} = \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \tag{C.1.2}$$

center of fibre A's right half-sphere to B's center of mass: $\overline{\mathbf{M_{A1}cm_{B1,\alpha}}} = \overline{\mathbf{M_{A1}cm_{B1,\alpha=0}}}\ \mathbf{R_y}$

center of mass of fibre B to collision point $\mathbf{P_\alpha}$: $\mathbf{cm_{B1,\alpha}} = \overline{\mathbf{M_{A1}cm_{B1,\alpha}}} + \mathbf{M_{A1}}$

center of fibre A's right half sphere to collision point $\mathbf{P_\alpha}$: $\overline{\mathbf{M_{A1}P_\alpha}} = \overline{\mathbf{M_{A1}P_{\alpha=0}}}\ \mathbf{R_y}$ (C.1.3)

center of mass of fibre A to collision point $\mathbf{P_\alpha}$: $\overline{\mathbf{cm_{A1}P_\alpha}} = \mathbf{M_{A1}} + \overline{\mathbf{M_{A1}P_\alpha}}$

center of mass of fibre B to collision point $\mathbf{P_\alpha}$: $\overline{\mathbf{cm_{B1}P_\alpha}} = \overline{\mathbf{M_{A1}P_\alpha}} - \overline{\mathbf{M_{A1}cm_{B1,\alpha}}}$

normal vector for collision – normal to tangent on half sphere in $\mathbf{P_\alpha}$: $\mathbf{e_{n,P_\alpha}} = \dfrac{\overline{\mathbf{M_{A1}P_\alpha}}}{\left|\overline{\mathbf{M_{A1}P_\alpha}}\right|}$

C.1.2 Post-Collisional Translational and Angular Velocities

Considering that the collision point $\mathbf{P_\alpha}$, the vectors from the center of mass to the collision point $\overline{\mathbf{cm_{A1}P_\alpha}}, \overline{\mathbf{cm_{B1}P_\alpha}}$ and the normal vector $\mathbf{e_{n,P\alpha}}$ to the collision plane are determined, the post-collisional values for the velocity and the angular velocity of both fibres can be calculated, according to an oblique, eccentric collision [60].

Note: Due to computation simplification for the contact point determination the fibres were considered to be spherocylinders. For the determination of the post-collisional velocities and angular velocities the fibres' moment of inertia tensor is assumed to be that of a spheroid. This is done to be consistent with the fibre-fluid interaction model.

In a collision of two bodies large forces occur at the contact points. In our work the assumption is made that the (finite) momentum change $\hat{\mathbf{I}} = \int \mathbf{F}\, dt$ caused by the collision force occurs in an infinitely small time increment $\Delta t \rightarrow 0$. This allows us to derive an analytical expression for post-collisional quantities.

We define a coefficient of restitution ε as the ratio of the post-collisional relative velocity (at the contact point) to the pre-collision relative velocity:

$$\varepsilon = \frac{\text{post-collision relative velocity}}{\text{pre-collision relative velocity}} \tag{C.1.4}$$

The coefficient of restitution ε varies between 0 (inelastic collision) and 1 (elastic collision) [60]. The connection between the coefficient of restitution and the spring stiffness and viscous damping coefficient is summarized in Chapter C.1.3.

In case that there is no friction at the contact point, the contact force **F** is normal to the collision plane, following $\mathbf{\hat{I}} = \hat{I}\,\mathbf{e}_{\mathbf{n,P\alpha}}$ with $\mathbf{e}_{\mathbf{n,P\alpha}}$ being the normal vector to the collision plane. To calculate the value $\hat{I}$, the coefficient of restitution ε and the velocities of the collision point at each fibre before $\mathbf{v}_{\mathbf{PA,1}}$, $\mathbf{v}_{\mathbf{PB,1}}$ and after the collision $\mathbf{v}_{\mathbf{PA,2}}$, $\mathbf{v}_{\mathbf{PB,2}}$ are used [60]:

$$\left(\mathbf{v}_{\mathbf{P_A,2}} - \mathbf{v}_{\mathbf{P_B,2}}\right)\cdot \mathbf{e}_{\mathbf{n,P_\alpha}} = -\varepsilon\left(\mathbf{v}_{\mathbf{P_A,1}} - \mathbf{v}_{\mathbf{P_B,1}}\right)\cdot \mathbf{e}_{\mathbf{n,P_\alpha}} \tag{C.1.5}$$

This means that the relative velocity of the fibres after the collision is proportional to the relative velocity before the collision, with the proportionality factor being ε. The initial (pre-collision) velocity of the collision point $\mathbf{P}_\alpha$ of each fibre is [61]:

$$\begin{aligned}\mathbf{v}_{\mathbf{P_A,1}} &= \mathbf{v}_{\mathbf{A,1}} + \boldsymbol{\omega}_{\mathbf{A,1}} \times \overline{\mathbf{cm}_{\mathbf{A,1}}\mathbf{P}_\alpha}\\ \mathbf{v}_{\mathbf{P}_B\mathbf{,1}} &= \mathbf{v}_{\mathbf{B,1}} + \boldsymbol{\omega}_{\mathbf{B,1}} \times \overline{\mathbf{cm}_{\mathbf{B,1}}\mathbf{P}_\alpha}\end{aligned} \tag{C.1.6}$$

The final (post-collisional) particle-centered translational and angular velocity of each fibre is [60]:

$$\begin{aligned}\mathbf{v}_{\mathbf{A,2}} &= \mathbf{v}_{\mathbf{A,1}} + \Delta\mathbf{v}_{\mathbf{A}} \qquad & \boldsymbol{\omega}_{\mathbf{A,2}} &= \boldsymbol{\omega}_{\mathbf{A,1}} + \Delta\boldsymbol{\omega}_{\mathbf{A}}\\ \mathbf{v}_{\mathbf{B,2}} &= \mathbf{v}_{\mathbf{B,1}} + \Delta\mathbf{v}_{\mathbf{B}} \qquad & \boldsymbol{\omega}_{\mathbf{A,2}} &= \boldsymbol{\omega}_{\mathbf{A,1}} + \Delta\boldsymbol{\omega}_{\mathbf{A}}\end{aligned} \tag{C.1.7}$$

The final (post-collisional) translational and angular velocity at the collision point $\mathbf{P}_\alpha$ of each fibre is [60]:

$$\begin{aligned}\mathbf{v}_{\mathbf{P_A,2}} &= \mathbf{v}_{\mathbf{A,1}} + \Delta\mathbf{v}_{\mathbf{A}} + \left(\boldsymbol{\omega}_{\mathbf{A,1}} + \Delta\boldsymbol{\omega}_{\mathbf{A}}\right)\times \overline{\mathbf{cm}_{\mathbf{A,1}}\mathbf{P}_\alpha}\\ \mathbf{v}_{\mathbf{P_B,2}} &= \mathbf{v}_{\mathbf{B,1}} + \Delta\mathbf{v}_{\mathbf{B}} + \left(\boldsymbol{\omega}_{\mathbf{B,1}} + \Delta\boldsymbol{\omega}_{\mathbf{B}}\right)\times \overline{\mathbf{cm}_{\mathbf{B,1}}\mathbf{P}_\alpha}\end{aligned} \tag{C.1.8}$$

Fibre A experiences a momentum change $\mathbf{\hat{I}}$**,** while fibre B experiences exactly the opposite momentum change $-\mathbf{\hat{I}}$. Thus, the integral momentum of the fibres remains constant [61]. The change of the translational and angular velocity can thus be defined as [60]:

$$\begin{aligned}\Delta\mathbf{v}_{\mathbf{A}} &= -\frac{\hat{I}\,\mathbf{e}_{\mathbf{n,p\alpha}}}{m_A} \qquad & \Delta\boldsymbol{\omega}_{\mathbf{A}} &= -\overline{\mathbf{cm}_{\mathbf{A,1}}\mathbf{P}_\alpha} \times \frac{\hat{I}\,\mathbf{e}_{\mathbf{n,p\alpha}}}{I_A}\\ \Delta\mathbf{v}_{\mathbf{B}} &= \frac{\hat{I}\,\mathbf{e}_{\mathbf{n,p\alpha}}}{m_B} \qquad & \Delta\boldsymbol{\omega}_{B} &= \overline{\mathbf{cm}_{\mathbf{B,1}}\mathbf{P}_\alpha} \times \frac{\hat{I}\,\mathbf{e}_{\mathbf{n,p\alpha}}}{I_B}\end{aligned} \tag{C.1.9}$$

I_A and I_B are the moment of inertia of the two fibres. In the present work the fibres are considered to be spheroids. The angular momentum for spheroids (for rotation around one of the two small axes, in our example this is the y^+-axis) is [62]:

$$I_A = I_B = \frac{1}{5} m \left(d_{Major}^{\ 2} + d_{Minor}^{\ 2} \right) \tag{C.1.10}$$

The missing information is $\hat{I}$. It can be determined by substitution of Eqn.(C.1.6). Eqn. (C.1.8) and Eqn.(C.1.9) in Eqn.(C.1.5), yielding:

$$\hat{I} = \frac{(1+\varepsilon)(\mathbf{v}_{\mathbf{P_A},1} - \mathbf{v}_{\mathbf{P_B},1}) \cdot \mathbf{e}_{\mathbf{n},\mathbf{P}\alpha}}{\frac{1}{m_A} + \frac{1}{m_B} + \left[\left(\overline{\mathbf{cm}_{\mathbf{A},1}\mathbf{P}_\alpha} \times \mathbf{e}_{\mathbf{n},\mathbf{P}\alpha} \right) \times \overline{\mathbf{cm}_{\mathbf{A},1}\mathbf{P}_\alpha} \frac{1}{I_A} + \left(\overline{\mathbf{cm}_{\mathbf{B},1}\mathbf{P}_\alpha} \times \mathbf{e}_{\mathbf{n},\mathbf{P}\alpha} \right) \times \overline{\mathbf{cm}_{\mathbf{B},1}\mathbf{P}_\alpha} \frac{1}{I_B} \right] \cdot \mathbf{e}_{\mathbf{n},\mathbf{P}\alpha}} \tag{C.1.11}$$

With Eqn. (C.1.11) and Eqn. (C.1.10), Eqn. (C.1.9) can be solved and the final (post-collisional) translational and angular velocity can be obtained [60].

C.1.3 Coefficient of Restitution

In our work a linear spring-dashpot model is used to resolve fibre-fibre contacts. Our model consists of a linear repulsive and a linear dissipative force. The fibre-fibre contact can thus be interpreted as a damped harmonic oscillator with a typical contact time of:

$$t_c = \frac{\pi}{\omega} \qquad \text{with: } \omega = \sqrt{\left(k / m_{12}\right) - \eta_0} \qquad \eta_0 = \frac{\gamma_0}{2 m_{12}} \qquad m_{12} = m_1 m_2 / \left(m_1 + m_2\right) \tag{C.1.12}$$

Here ω is the eigenfrequency of the contact, k is the spring stiffness, η_0 the rescaled damping coefficient, γ_0 the viscous damping coefficient, m_{12} the reduced mass and m_1 and m_2 the mass of particle *1* and *2*. Furthermore, the coefficient of restitution ε (defined above) can be determined as [26]:

$$\varepsilon_d^{\ 0} = \exp\left[-\eta_0 \, t_c\right] \tag{C.1.13}$$

Note, when using a linear spring-dashpot model, the coefficient of restitution is only a function of the material parameters k and γ_0, and hence is constant. Furthermore, we note that Schwager and Pöschel [63] enhanced the calculation of the coefficient of restitution by taking into account that a non-cohesive system can never exhibit attractive forces: when using the above expression, the interaction force between the two particles is assumed to become negative. This means that an attraction force occurs even though an exclusive repulsive interaction is assumed (as is done in our simulations). To avoid such an artifact, the following equations for ε_d were developed:

$$\varepsilon_d = \begin{cases} \exp\left[-\dfrac{\beta}{\omega}\left(\pi - \arctan\dfrac{2\beta\omega}{\omega^2-\beta^2}\right)\right] & for \quad \beta < \dfrac{\omega_0}{\sqrt{2}} \\ \exp\left[-\dfrac{\beta}{\omega}\arctan\dfrac{2\beta\omega}{\omega^2-\beta^2}\right] & for \quad \beta \in \left[\dfrac{\omega_0}{\sqrt{2}}, \omega_0\right] \\ \exp\left[-\dfrac{\beta}{\Omega}\ln\dfrac{\beta+\Omega}{\beta-\Omega}\right] & for \quad \beta > \omega_0 \end{cases} \qquad \text{(C.1.14)}$$

with $\beta = \eta_0$, $\omega_0 = (k / m_{12})^{0.5}$ and $\Omega = (\beta^2 - \omega_0^2)^{0.5}$. However, the difference between ε_d and ε_d^0 is small as long as ε_d is well above ~ 0.7 [63].

C.1.4 Energy Conservation

Since the effect of gravity is not modeled, the initial (pre-collisional) energy and the final (post-collisional) total energy of the system is equal to the sum of the potential energy (of the spring elements on the contact model) and the kinetic energy. The latter can be divided into translational energy and rotational energy:

$$\begin{aligned} E_{tot} &= \sum_{i=A}^{B} E_{kin,i} = \sum_{i=A}^{B} E_{Trans,i} + \sum_{i=A}^{B} E_{Rot,i} \\ E_{Trans,i} &= \frac{1}{2} m_i \, \mathbf{v_i}^2 \\ E_{Rot,i} &= \frac{1}{2} I_i \, \boldsymbol{\omega}_\mathbf{i}^2 \end{aligned} \qquad \text{(C.1.15)}$$

I_i is the moment of inertia around the y^+-axis, $\mathbf{v_i}$ the velocity and $\boldsymbol{\omega}_\mathbf{i}$ the angular velocity of the fibre *i*. Note, to square the vectors the dot product of the vector needs to be considered [64].

In case the collision is fully elastic, and in the absence of friction (i.e., the damping = 0) the total energy of the system stays constant [61].

C.1.5 Momentum Conservation

The momentum is divided into linear and angular momentum. In case of a fibre-fibre collision the linear and the angular moment are conserved [61].

The linear momentum is then defined as [64]:

$$\sum_{i=A}^{B} \mathbf{p_i} = \sum_{i=A}^{B} m \, \mathbf{v_i} = const. \qquad \text{(C.1.16)}$$

and the angular momentum is defined as [64]:

$$\sum_{i=A}^{B} \mathbf{L_i} = \sum_{i=A}^{B} \mathbf{r}_i \times \mathbf{p}_i = \sum_{i=A}^{B} \mathbf{r}_i \times m_i \, \mathbf{v}_i = \sum_{i=A}^{B} I_i \, \boldsymbol{\omega}_i = const. \tag{C.1.17}$$

I_i is the moment of inertia around the y^+-axis, $\mathbf{v_i}$ the velocity, $\boldsymbol{\omega}_\mathbf{i}$ the angular velocity and r_i the vector from the origin to the center of mass of fibre i.

C.1.6 Test Case Results

The parameters used for the test case are listed in Table 16. In Table 17 the minimum/maximum number of time steps for resolving a single fibre-fibre contact with the used stiffness and damping parameters are listed. The spring stiffness and the damping coefficient are chosen such that the minimum number of time steps for a single fibre-fibre contact is >50. The results from the fibre-fibre interaction test case are shown in Fig. 57. The results for the velocity, the angular velocity, the total kinetic energy and the total linear momentum of both fibres, obtained from the numerical simulation and the analytical solution, are compared. The results show excellent agreement and hence the implemented code is assumed to be correct.

Table 16: Test Case Parameters.

Test Case Angles			
α	[0 15 30 45 60 75 90]		
Fibre Parameter			
d_{Major}	1.00	v_A	[0 0 0]
d_{Minor}	0.10	ω_A	[0 0 0]
ρ	1.27	v_B	[-10 0 0]
x	0.60	ω_B	[0 0 0]
Interaction Parameters			
stiffnessNormal	$2.0 \cdot 10^4$	roughFact	$1.0 \cdot 10^{-2}$
dampingNormal	$1.0 \cdot 10^{-2}$	roughStiffFact	$1.0 \cdot 10^{-3}$
stiffnessTang	0.0	frictionCoeff	0.0
dampingTang	0.0	liquidDynViscocity	0.0
DEM Parameter			
Δt_{DEM}	$1.0 \cdot 10^{-6}$		
$t_c/\Delta t_{DEM}$	50		

Table 17: Test Case min./max. contact time steps for fibre-fibre interaction.

$t_c/\Delta t_{DEM}$ =50	m_{ij}	k/m_{ij}	$(\gamma_0/2/m_{ij})^2$	ω	t_c	$\Delta t_{DEM.calc}$	$t_c/\Delta t_{DEM}$
AR2-AR2	$5.32 \cdot 10^{-6}$	$3.76 \cdot 10^{9}$	$8.83 \cdot 10^{5}$	$6.13 \cdot 10^{4}$	$5.12 \cdot 10^{-5}$	$1.02 \cdot 10^{-6}$	51.24
AR20-AR20	$5.32 \cdot 10^{-5}$	$3.76 \cdot 10^{8}$	$8.83 \cdot 10^{3}$	$1.94 \cdot 10^{4}$	$1.62 \cdot 10^{-4}$	$3.24 \cdot 10^{-6}$	162.03

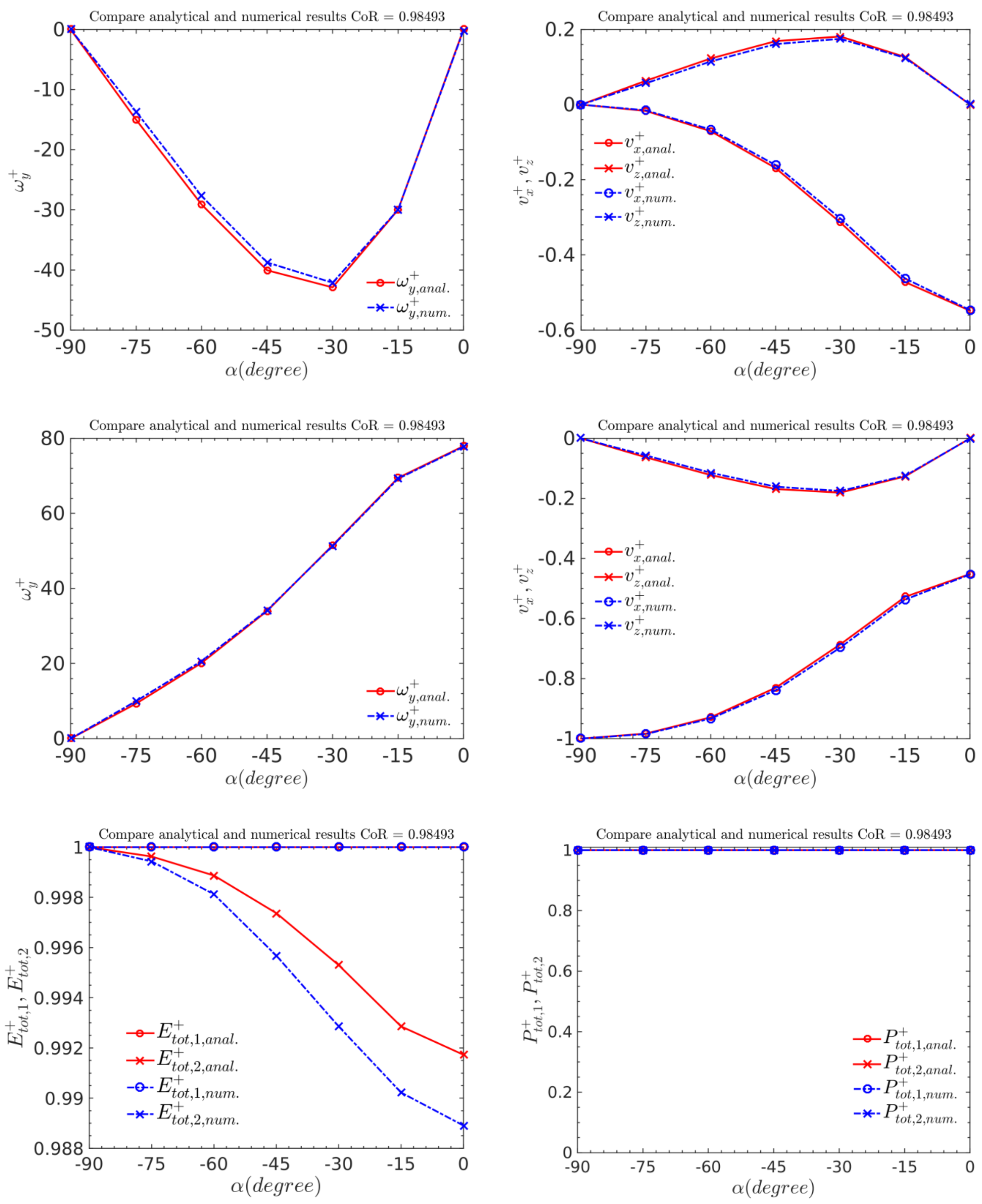


Fig. 58: Results fibre-fibre interaction code testing: analytical vs. numerical solution. Top row: fibre 1 (=A). Left: angular velocity around y^+-axis. Right: velocity in x^+- and z^+-direction. Middle row: fibre 2 (=B). Left: angular velocity around y^+-axis. Right: velocity in x^+- and z^+-direction. Bottom row: Left: total energy of both fibres. Right: linear momentum of both fibres.

C.2 Fibre-Flock Generation Pre-Processing

Assumptions:

- Fibres are considered to be spherocylinder
- The density of the fibres is constant (1.27)
- Each fiber class which is based on the fibre length d_{Major} has only one d_{Minor} value.

C.2.1 LIGGGHTS® input

The input dataset is given by:

Δt_{DEM}	DEM time step
$\Delta t_{compact}$ / Δt_{DEM}	deform every this many DEM time steps
k_{norm}	spring stiffness in normal direction
k_{tang}	spring stiffness in tangential direction, set to zero, not part of the model
$\gamma_{0.norm}$	viscous damping coefficient in normal direction; small damping removes kinetic energy gently, in case the damping coefficient is set to zero the coefficient of restitution is zero.
$\gamma_{0.tang}$	viscous damping coefficient in tangential direction, set to zero
DefMod	gives the type of deformation it can either be 1 for uni-axial (in *x*-direction) or 3 for tri-axial.
$\Delta Q_{r.i}$	the fibre distribution with its $d_{Minor,i}$ and $d_{Major,i}$ values
r	basis of the distribution (either number or volume based)
box_{final}	the final box dimensions (x^+, y^+, z^+) e.g. [1 1 1].
box_{init}	the initial box dimensions (x^+, y^+, z^+) e.g. [10 10 10].
φ_{Fib}	fibre volume fraction in the reduced final box (post-deformation)
ρ_{Fib}	density of the fibres
ρ_{Fluid}	fluid density
$k_{overlap}$	maximum overlap of two fibres at a single deformation
μ	friction coefficient; the ratio of the friction force and the normal force.
η_{fluid}	Dynamic viscosity of the fluid is needed for the lubrication model for fibre-fibre interaction, if it is set to zero, there is no lubrication

Roughness factor representing the fraction of d_{Minor} for roughness layer of fibre to model fibrils on fibre (it cannot be set to zero)

RoughnessStiffFact fraction of the fibres normal spring stiffness used to calculate the repelling force due to the roughness layers contact

C.2.2 General Definitions

Aspect Ratio [31]

$$AR_i = \frac{d_{Major,i}}{d_{Minor,i}} \tag{C.2.1}$$

Fibre Distributions [48]

r = 0 … number based distribution *r = 3* … volume based distribution

$$\Delta Q_{r=0,i} = \frac{n_{Fib,tot,i}}{n_{Fib,tot}} \qquad \Delta Q_{r=3,i} = \frac{m_{Fib,tot,i}}{m_{Fib,tot}} \tag{C.2.2}$$

Consistency

$$\varphi_{Fib} = \frac{m_{Fib,tot}}{m_{Fib,tot} + m_{Fluid}} \tag{C.2.3}$$

In case the densities of fluid and fibre are unity, the consistency is equal the fibre volume fraction.

C.2.3 Pre-Calculations

Initial (pre-deformation) and reduced final (post-deformation) box

volume of initial (pre-deformation) box

$$V_{init} = box_{init,X}\, box_{init,Y}\, box_{init,Z} \tag{C.2.4}$$

volume of reduced final (post-deformation) box

$$V_{red-final} = \left(box_{final,X} - d_{Major,\max}\right)\left(box_{final,Y} - d_{Major,\max}\right)\left(box_{final,Z} - d_{Major,\max}\right) \tag{C.2.5}$$

Fibre Volume Fraction in the reduced final (post-deformation) Box

$$\varphi_{Fib} = \frac{V_{Fib,tot}}{V_{red-final}} \rightarrow V_{Fib,tot} = \varphi_{Fib}\, V_{red-final} \tag{C.2.6}$$

Fibre Volume and Mass

volume of a single fibre in class *i* (= volume of an ellipsoid)

$$V_{Fib,i} = \frac{d_{Major,i}^{3}\,\pi}{6\,AR_i^2} \tag{C.2.7}$$

mass of all fibres in the system

$$m_{Fib,tot} = \rho_{Fib}\, V_{Fib,tot} \tag{C.2.8}$$

mass of single fibre in class *i*

$$m_{Fib,i} = \rho_{Fib}\, V_{Fib,i} \tag{C.2.9}$$

Fluid volume and mass

volume of fluid

$$V_{Fluid} = V_{red-final} - V_{Fib,tot} = (1-\varphi_{Fib})\,V_{red-final} \tag{C.2.10}$$

mass of fluid

$$m_{Fluid} = \rho_{Fluid}\,V_{Fluid} \tag{C.2.11}$$

C.2.4 Number of Fibres in each Class *i*

$\Delta Q_{r=0,i}$ is given by

$$V_{Fib,tot} = \sum_{i}^{m} n_{Fib,tot,i}\,V_{Fib,i} \qquad /:n_{Fib,tot}$$

$$\frac{V_{Fib,tot}}{n_{Fib,tot}} = \sum_{i}^{m}\frac{n_{Fib,tot,i}}{n_{Fib,tot}}V_{Fib,i} \tag{C.2.12}$$

$$n_{Fib,tot} = \frac{V_{Fib,tot}}{\sum_{i}^{m}\Delta Q_{r=0,i}\,V_{Fib,i}}$$

$$\rightarrow\; n_{Fib,tot,i} = \Delta Q_{r=0,i}\;n_{Fib,tot}$$

$\Delta Q_{r=3,i}$ is given

$$m_{Fib,tot,i} = \Delta Q_{r=3,i}\,m_{Fib,tot}$$

$$\rightarrow n_{Fib,tot,i} = \frac{m_{Fib,tot,i}}{m_{Fib,i}} \tag{C.2.13}$$

Note, since the fibre number is an integer value the fibre fraction (which is specified by the user) might change due to round-off errors. The above calculations are based on Stieß' [48] fundamental explanations of particle distribution.

C.2.5 Initial Fibre Position and Orientation

At the beginning of the simulation the fibres are placed randomly (random position of center of mass and random orientation) over the initial simulation box domain. The positioning of the fibres is done in a MATLAB/OCTAVE script using random numbers sampled from a uniform distribution by using the `rand()` function. The random fibre orientation is done in LIGGGHTS® using `quat/random(666)`.

The fibre orientation is basically a vector representing the direction of the x^{+}-axis in the fibre's coordinate system, and pointing in the direction of the fibres main axis (d_{Major}). The fibre can rotate in all possible direction, i.e., a point on the surface of a sphere needs to be randomly sampled. To sample random points on the surface of a sphere, it is not accurate to sample the spherical coordinates (ϕ, θ) form a uniform distribution with $\theta \epsilon [0, 2\pi]$ and $\phi \epsilon [0, \pi]$: clearly, this would generate higher concentration of points around the poles. The correct approach is to sample two random numbers u and v [0, 1], and then determine the spherical coordinates via the transformation $\theta = 2 \pi u$ and $\phi = cos^{-1}(2v\text{-}1)$. Fig. 59 shows the results of 2e4 uniformly distributed points on the surface of a unit sphere and the distribution of the azimuthal and polar angle of these points [65].

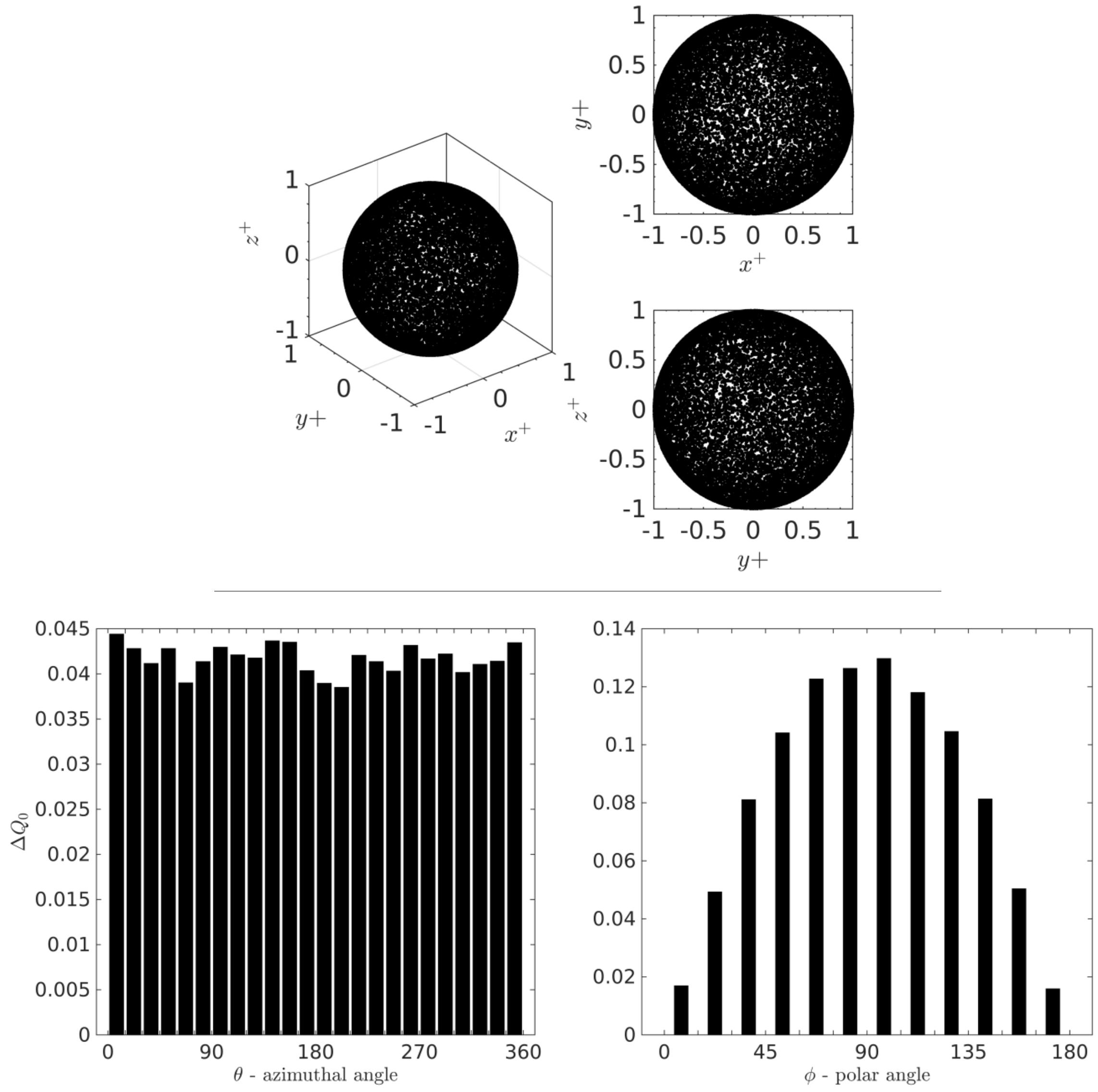


Fig. 59: 2e4 uniformly distributed points on the surface of a unit sphere. All classes have the same width of 15 degree. Top: points on the surface of the sphere. Left: azimuthal angle distribution. Right: polar angle distribution.

C.2.6 Time Step Parameters

In the simulation three different time parameters need to be considered: the characteristic response t_c time, the DEM time step Δt_{DEM} and the deformation/compaction time $t_{compact}$. The characteristic contact time t_c can be determined according to Luding [26]:

$$t_c = \frac{\pi}{\omega} \qquad \omega = \sqrt{\frac{k}{m_{12}} - \eta_0{}^2} \qquad \eta_0 = \frac{\gamma_0}{2m_{12}} \qquad m_{12} = \frac{m_1 m_2}{m_1 + m_2} \tag{C.2.14}$$

k represents the spring stiffness, m_{12} the reduced mass, η_0 the reduced viscous damping coefficient and γ_0 the viscous damping coefficient.

To guarantee a stable simulation the DEM time step Δt_{DEM} is kept a lot smaller than t_c, $\Delta t_{DEM}= t_c/50$. Deformation occurs after at least every 50 DEM time steps: $t_{compact} = \Delta t_{DEM}/50$.

C.2.7 Deformation Rate

At each deformation the simulation box is deformed with a constant deformation rate:

$$\varepsilon_{deform} = \frac{\Delta L}{L_t} = \frac{L_t - L_{t+\Delta t}}{L_t} = 1 - \frac{L_{t+\Delta t}}{L_t} \tag{C.2.15}$$

The total deformation rate is:

$$\varepsilon_{tot} = \frac{\Delta L}{L_0} = \frac{L_0 - L_{end}}{L_0} = 1 - \frac{L_{end}}{L_0} \tag{C.2.16}$$

C.2.8 Fibre Overlap

Considering two of the longest fibres (d_{Major}) touching each other, the fibres' overlap at each deformation is $\varepsilon_{deform} \cdot d_{Major}$. This overlap has to be smaller than a certain overlap factor $k_{overlap} \cdot d_{Minor}$ to avoid that fibres being cut off (or penetrated) during one deformation step.

$$k_{overlap}\, d_{Minor} \geq \varepsilon_{deform}\, d_{Major} \qquad \rightarrow \qquad \varepsilon_{deform} \leq k_{overlap} \frac{d_{Minor}}{d_{Major}} \tag{C.2.17}$$

C.2.9 Trate Input Parameter R [56]

The length of the simulation box after a certain time can be determined by:

$$L_t = L_o \exp[R\, \Delta t] \tag{C.2.18}$$

With L_0 being the original length, L_t the length after Δt, R the *trate* value and Δt the elapsed time between L_0 and L_t. Knowing the elapsed time for a single deformation Δt_{deform} and the deformation rate ε_{deform} for a single deformation step, R can be calculated via:

$$\varepsilon_{deform} = 1 - \frac{L_{t+\Delta t}}{L_t} = 1 - \exp\left[R\ \Delta t_{deform}\right] \rightarrow R = \frac{\ln\left(1-\varepsilon_{deform}\right)}{\Delta t_{deform}} \tag{C.2.19}$$

Note, the first approach was a deformation with a constant ΔL for each deformation step. This approach caused computational problems, particularly at late times of the deformation process. This is due to the fact that the deformation rate ε_{deform} increased during the deformation process. At the end of the deformation process, when the fibres are already very close to each other and interact frequently, an increasing ε_{deform} might result in unrealistic fibre cut off and fibre-fibre penetration.

C.2.10 Run Input Parameter

The number of DEM time steps that are necessary for reaching a certain box size due to deformation can be determined by:

$$\#_{DEM} = \frac{\ln\left(\frac{L_{End}}{L_0}\right)}{R\,\Delta t_{DEM}} = \frac{\ln\left(1-\varepsilon_{tot}\right)}{R\,\Delta t_{DEM}} \tag{C.2.20}$$

Since the number of DEM time steps needs to be an integer number, it is necessary to calculate the compact time steps, round the number to the next integer and determine the necessary DEM time steps via:

$$\#_{DEM_{\text{Real}}} = \text{roundup}\left[\frac{\ln\left(1-\varepsilon_{tot}\right)}{R\,\Delta t_{DEM}} \frac{\Delta t_{DEM}}{\Delta t_{compact}}\right] \frac{\Delta t_{compact}}{\Delta t_{DEM}} \tag{C.2.21}$$

C.3 Fibre-Flock Generation Post Processing

To analyze the behavior of the fibres during the compaction step, or later in the T-junction simulation, the fibre orientation and the angular velocity of each fibre j is investigated by post-processing LIGGGHTS® simulation output data.

C.3.1 LIGGGHTS® Output Data

The LIGGGHTS® output file (dump-file) contains amongst others the following information:

- The position vector of the fibre j $\quad \left[\mathbf{x}_\mathbf{j}^+\right]_\mathbf{B} = \left[x_j^+ \;\; y_j^+ \;\; z_j^+\right]_B$
- The orientation vector of the fibre j $\quad \left[\mathbf{fex}_\mathbf{j}\right]_\mathbf{B} = \left[fex1_j \;\; fex2_j \;\; fex3_j\right]_B$ (C.3.1)
- The angular velocity vector of the fibre j $\quad \left[\boldsymbol{\omega}_\mathbf{j}\right]_\mathbf{B} = \left[\omega_{x,j} \;\; \omega_{y,j} \;\; \omega_{z,j}\right]_B$

All vectors in the dump files refer to the global coordinate system **B** ($\mathbf{e_x}$, $\mathbf{e_y}$, $\mathbf{e_z}$).

C.3.2 Dividing Fibres into the 4 Sections according to their Position

From the LIGGGHTS® output file the position vector $[\boldsymbol{x_j}]_\mathbf{B}$ (based on the global coordinate system) is given, so that the classification of the fibres to the sections can be realized by following:

Table 18: Fibre sectioning

section 1				
	$[y^+_{min}]_B < [y^+_j]_\mathbf{B} \le 0$	and	$[z^+_{min}]_B \le [z^+_j]_\mathbf{B} < -\lvert[y^+_i]_\mathbf{B}\rvert$	or
	$0 \le [y^+_j]_\mathbf{B} \le [y^+_{max}]_B$	and	$[z^+_{min}]_B \le [z^+_j]_\mathbf{B} \le -\lvert[y^+_i]_\mathbf{B}\rvert$	
section 2				
	$0 \le [y^+_j]_\mathbf{B} \le [y^+_{max}]_B$	and	$-\lvert[y^+_i]_\mathbf{B}\rvert < [z^+_j]_\mathbf{B} \le 0$	or
	$0 \le [y^+_j]_\mathbf{B} \le [y^+_{max}]_B$	and	$0 \le [z^+_j]_\mathbf{B} \le \lvert[y^+_i]_\mathbf{B}\rvert$	
section 3				
	$0 \le [y^+_j]_\mathbf{B} \le [y^+_{max}]_B$	and	$\lvert[y^+_i]_\mathbf{B}\rvert \le [z^+_j]_\mathbf{B} \le [z^+_{max}]_B$	or
	$[y^+_{min}]_B \le [y^+_j]_\mathbf{B} \le 0$	and	$\lvert[y^+_i]_\mathbf{B}\rvert < [z^+_j]_\mathbf{B} \le [z^+_{max}]_B$	
section 4				
	$[y^+_{min}]_B \le [y^+_j]_\mathbf{B} \le 0$	and	$0 \le [z^+_j]_\mathbf{B} < \lvert[y^+_i]_\mathbf{B}\rvert$	or
	$[y^+_{min}]_B \le [y^+_j]_\mathbf{B} \le 0$	and	$-\lvert[y^+_i]_\mathbf{B}\rvert \le [z^+_j]_\mathbf{B} \le 0$	

$[y^+_{min}]_B$ and $[y^+_{max}]_B$ are the minimum and maximum position in the y^+-direction and $[z^+_{min}]_B$ and $[z^+_{max}]_B$ are the minimum and maximum position in the z^+-direction.

Note, each section triangle has two boundary lines to the neighboring sections. For the fibre section classification the boundary line on the left of each section triangle belongs to the section, as indicated by the small grey arrows in Fig. 9. Fibres at the origin belong to the first section.

C.3.3 Fibre Orientation

Now, that the fibres are divided into 4 cross section sections the fibre orientation vector $[\mathbf{fex_j}]_\mathbf{B}$. based on the global coordinate system, needs to be transformed to the new basis $\mathbf{B_i''}$ ($\mathbf{e_{x,i}''}$, $\mathbf{e_{y,i}''}$, $\mathbf{e_{z,i}''}$) of each section *i.*

C.3.3.1 Change of Basis: B → B_i''

B ($\mathbf{e_x}$, $\mathbf{e_y}$, $\mathbf{e_z}$) representing the global coordinate system used by LIGGGHTS® and $\mathbf{B_i''}$ ($\mathbf{e_{x,i}''}$, $\mathbf{e_{y,i}''}$, $\mathbf{e_{z,i}''}$) representing the transformed coordinate systems, which differs in each section *i*.

The orientation of the orthogonal coordinate system in section 1 ($\mathbf{e_{x,1}''}$, $\mathbf{e_{y,1}''}$, $\mathbf{e_{z,1}''}$) is equivalent to the orientation of the orthogonal global system ($\mathbf{e_x}$, $\mathbf{e_y}$, $\mathbf{e_z}$). For the other sections (*i* =2, 3, 4) the global coordinate system needs to be turned in $\pi/2\cdot(i\text{-}1)$ steps around the x^+-axis ($\alpha_{x,i}$).

Table 19: Change of basis B → B_i'': rotation angle for each section *I*, around the global x^+-axis.

section	rotation angle
section 1	$\alpha_{x1} = 0$
section 2	$\alpha_{x2} = \pi/2$
section 3	$\alpha_{x3} = \pi$
section 4	$\alpha_{x4} = 3/2\pi$

The changes of basis **B** → $\mathbf{B}_i''$ are orthogonal coordinate transformations, realized by 4 passive rotations (see chapter C.3.5.2) around the x^+-axis unit vector $[\mathbf{e_x}]_\mathbf{B} = [1, 0, 0]_\mathbf{B}$ of the global coordinate system. The rotation angles $\alpha_{x,i}$ are shown in Table 19.

For the unit fibre orientation vector the transformation in the section based coordinates system can be calculated by:

$$\left[\mathbf{fex_j}\right]_{\mathbf{B_i''}} = \left[\mathbf{D_i}\right]_\mathbf{B}^{\mathbf{B_i''}} \left[\mathbf{fex_j}\right]_\mathbf{B} \tag{C.3.2}$$

C.3.4 Angular Velocity

The angular velocity vector available in the LIGGGHTS® dump-files $[\boldsymbol{\omega}_\mathbf{j}]_\mathbf{B}=[\omega_{x,j}\ \omega_{y,j}\ \omega_{z,j}]_\mathbf{B}$ describes the rotation of the fibre around the global x^+-, y^+- and z^+-axis.

C.3.4.1 Change of Basis: B → B_i'' → $B_{j,i}'$

To be able to analyze the fibre angular velocity in each section *i* the angular velocity needs to be transformed from $[\boldsymbol{\omega}_j]_B$ to $[\boldsymbol{\omega}_j]_{Bi''}$ as summarized in chapter C.3.3.1.

$$\left[\boldsymbol{\omega}_j\right]_{B_i''} = \left[\mathbf{D}_i\right]_B^{B_i''} \left[\boldsymbol{\omega}_j\right]_B \tag{C.3.3}$$

Since the rotation of the fibre around its own x^+-axis $[\mathbf{e}_x]_{Bi''}$ (i.e., “log rolling”) is not of interest, the angular velocity needs to be corrected. To compute the corrected angular velocity in the $\mathbf{B}_i''$ coordinate system, the angular velocity $[\boldsymbol{\omega}_j]_{Bi''}$ was transformed to the fibre’s coordinate system $\mathbf{B}_{j.i}'$ ($\mathbf{e}_{x,j,i}'$, $\mathbf{e}_{y,j,i}'$, $\mathbf{e}_{z,j,i}'$), corrected in this coordinate system, and then transformed back to the $\mathbf{B}_i''$ coordinate system.

$$\left[\boldsymbol{\omega}_j\right]_{B_{j,i}'} = \left[\mathbf{D}_j\right]_{B_i''}^{B_{j,i}'} \left[\boldsymbol{\omega}_j\right]_{B_i''} \tag{C.3.4}$$

The fibre’s coordinates system’s x^+-axis points exactly into the direction of the orientation vector $[\mathbf{fex}_j]_{Bi''}$. In order to compute the rotation axis $[\mathbf{x}_{rot,j}]_{Bi''}$, and the rotation angle α_j for the transformation $\mathbf{B}_i''$ → $\mathbf{B}_{j,i}'$**,** the following needs to be considered:

In the transformation the unit basis vector in x^+-direction $[\mathbf{e}_x]_{Bi''}$ of the coordinate system $\mathbf{B}_i''$ and the unit orientation vector of the fibre *j* $[\mathbf{fex}_j]_{Bi''}$ lie on a 2D – plane. The angle α_j between those two vectors on the plane is the rotation angle necessary for the coordinate transformation. The normal vector to this plane represents the rotation angle $[\mathbf{x}_{rot,j}]_{Bi''}$ (see Fig. 60).

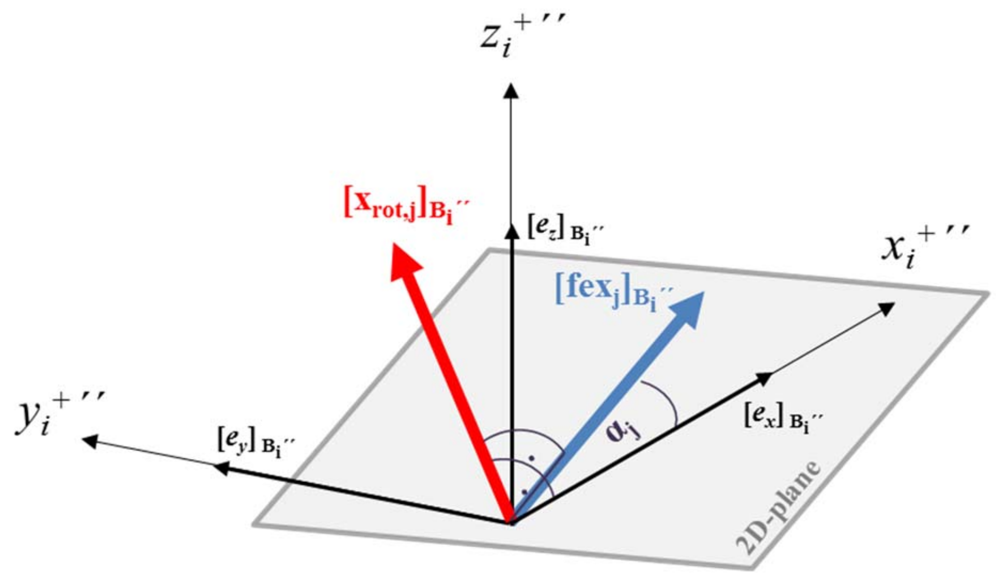


Fig. 60: Coordinate Transformation B_i'' → $B_{j,i}'$.

The rotation angle and the rotation axis can be determined by:

$$\alpha_j = \arccos\left(\frac{\left[\mathbf{fex_j}\right]_{\mathbf{B_i}''} \times \left[\mathbf{e_x}\right]_{\mathbf{B_i}''}}{\left\| \left[\mathbf{fex_j}\right]_{\mathbf{B_i}''} \times \left[\mathbf{e_x}\right]_{\mathbf{B_i}''} \right\|} \right)$$
$$\mathbf{x_{rot,j}} = \frac{\left[\mathbf{fex_j}\right]_{\mathbf{B_i}''} \times \left[\mathbf{e_x}\right]_{\mathbf{B_i}''}}{\left\| \left[\mathbf{fex_j}\right]_{\mathbf{B_i}''} \times \left[\mathbf{e_x}\right]_{\mathbf{B_i}''} \right\|} \qquad \text{(C.3.5)}$$

C.3.4.2 Reduction of the angular velocity

In the $\mathbf{B_{j.i}}'$ coordinate system the angular velocity gets reduced by its x^+-component. $[\mathbf{e_{x,j}}]_{\mathbf{Bj,i}'}$ is the unit base vector in x^+ – direction in the fibre's coordinate system.

$$\left[\boldsymbol{\omega}_{\mathbf{red,j}}\right]_{\mathbf{B_{j,i}}'} = \left[\boldsymbol{\omega}_{\mathbf{j}}\right]_{\mathbf{B_{j,i}}'} - \left(\left[\boldsymbol{\omega}_{\mathbf{j}}\right]_{\mathbf{B_{j,i}}'} \cdot \left[\mathbf{e_{x,j}}\right]_{\mathbf{B_{j,i}}'} \right) \cdot \left[\mathbf{e_{x,j}}\right]_{\mathbf{B_{j,i}}'} \qquad \text{(C.3.6)}$$

C.3.4.3 Transformation from Basis $B_{j,i}'$ to B_i''

For further analyses the corrected angular velocity is retransformed in the sections' coordinate systems. By using the same general transformation Matrix $\left[\mathbf{D_j}\right]_{\mathbf{B_i}''}^{\mathbf{B_{j,i}}'}$ as by the transformation from the sections' coordinate systems to the fibers' coordinate system, but transposed.

$$\left[\boldsymbol{\omega}_{\mathbf{red,j}}\right]_{\mathbf{B_i}''} = \left(\left[\mathbf{D_j}\right]_{\mathbf{B_i}''}^{\mathbf{B_{j,i}}'} \right)^{\mathbf{T}} \left[\boldsymbol{\omega}_{\mathbf{red,j}}\right]_{\mathbf{B_{j,i}}'} \qquad \text{(C.3.7)}$$

C.3.5 Mathematical Basics

C.3.5.1 Cartesian-to-Polar Coordinate Transformation [59]

For the conversion between the Cartesian and the polar coordinate system the following equations have been used. Note, in case the fibre's orientation $\mathbf{[fex_j]_{Bi''}}$ is equal to the $[z]_{\mathbf{Bi''}}$ - axis ($[x_j]_{\mathbf{Bi''}} = 0$ and $[y_j]_{\mathbf{Bi''}} = 0$) the orientation of the fibre is mathematically undefined – division by zero! Fibres with such an orientation are defined to have an azimuth angle of $\pi/2$. (see last if condition for Θ)

$$r = \sqrt{\left[x_j\right]_{B_i''}^{\;2} + \left[y_j\right]_{B_i''}^{\;2} + \left[z_j\right]_{B_i''}^{\;2}} \tag{C.3.8}$$

$$\left[\theta_j\right]_{B_i''} = \begin{cases} \arctan\dfrac{\left[y_j\right]_{B_i''}}{\left[x_j\right]_{B_i''}} & \text{if } \left[x_j\right]_{B_i''} > 0 \\ \arctan\dfrac{\left[y_j\right]_{B_i''}}{\left[x_j\right]_{B_i''}} + \pi & \text{if } \left[x_j\right]_{B_i''} < 0 \quad \left[y_j\right]_{B_i''} \geq 0 \\ \arctan\dfrac{\left[y_j\right]_{B_i''}}{\left[x_j\right]_{B_i''}} - \pi & \text{if } \left[x_j\right]_{B_i''} < 0 \quad \left[y_j\right]_{B_i''} < 0 \\ +\dfrac{\pi}{2} & \text{if } \left[x_j\right]_{B_i''} = 0 \quad \left[y_j\right]_{B_i''} > 0 \\ -\dfrac{\pi}{2} & \text{if } \left[x_j\right]_{B_i''} = 0 \quad \left[y_j\right]_{B_i''} < 0 \\ +\dfrac{\pi}{2} & \text{if } \left[x_j\right]_{B_i''} = 0 \quad \left[y_j\right]_{B_i''} = 0 \end{cases} \tag{C.3.9}$$

$$\left[\varphi_j\right]_{B_i''} = \cos^{-1}\left(\left[z_j\right]_{B_i''}\right) \tag{C.3.10}$$

C.3.5.2 General Rotation Matrix [64]

The general rotation matrix ***D*** allows a rotation of a vector or a coordinate system by any angle α around any rotation axis $[\boldsymbol{x_{rot}}]_{\mathbf{B}} = [u, v, w]_{\mathrm{B}}$. It is defined by:

$$\mathbf{D} = \begin{bmatrix} c+(1-c)u^2 & (1-c)uv+sw & (1-c)uw-sv \\ (1-c)uv-sw & c+(1-c)v^2 & (1-c)vw-su \\ (1-c)uw+sv & (1-c)vw-su & c+(1-c)w^2 \end{bmatrix} \qquad \text{(C.3.11)}$$

with $c = cos\ (\alpha)$ and $s{=}sin(\alpha)$

Active Rotation

In case a vector $[\boldsymbol{a}]_{\mathbf{B}}$ is rotated in a fixed coordinate system **B,** the new vector $[\boldsymbol{a}']_{\mathbf{B}}$ can be calculated via:

$$[\mathbf{a}']_{\mathbf{B}} = \mathbf{D}^{\mathbf{T}} \cdot [\mathbf{a}]_{\mathbf{B}} \qquad \text{(C.3.12)}$$

Passive Rotation

In case of a change of the basis **B** → **B′**, i.e., the coordinate system rotates but the vector is fixed in the coordinate system, the new vector coordinates referring to the new basis **B′**, can be determined via:

$$[\mathbf{a}]_{\mathbf{B}'} = \mathbf{D} \cdot [\mathbf{a}]_{\mathbf{B}} \qquad \text{(C.3.13)}$$

Note, an orthogonal coordinate transformation is a passive transformation, meaning that the values of all points and vectors in the system are only changing in respect to the changing basis. However, the position and orientation of the points and vectors does not change.

Appendix D CFD-DEM Simulation

D.1 Jeffery Orbit

Jeffery [45] derived an equation for the rotation rate of ellipsoids in a linear shear flow:

$$\dot{\gamma}_p = \frac{2\pi}{t^+} = \frac{\dot{\boldsymbol{\gamma}}}{AR + \frac{1}{AR}} \tag{D.1.1}$$

with t^+ the orbit period, AR the aspect ratio (d_{Major}/d_{Minor}) and the shear rate $\dot{\gamma}$.

D.2 Velocity Profile in a Square Cross Section

According to Tamayol et al. [46] the velocity profile of a laminar flow in a square cross section can be determined by:

$$\mathbf{u} = \mathbf{u}^+ \cdot U_{\max}$$

$$\mathbf{u}^+ = \left[1 - \frac{\eta}{4A_1} + \sum_{i=1}^{\infty} \frac{C_i}{A_1}\left(\eta^{mi}\cos\left(m\Theta\right)\right)\right] \qquad \eta = \frac{\mathrm{r}\tan\left(\frac{\pi}{m}\right)}{s} \tag{D.2.1}$$

$$A_1 = 0.247 + \frac{0.767}{m^2} \qquad C_1 = \frac{1}{6.01 - 3.12m - 0.965m^2}$$

The used parameters are: $U_{\max}$ = 2.08 (value taken from integrating **u** over the whole cross section), and C_2=-1.096·10^{-3}, determined from the no-slip condition at the wall.

D.3 Shear Rate

The shear rate can be determined via [66]:

$$\dot{\gamma} = \sqrt{2 \cdot tr\left(\mathbf{D}^2\right)} \tag{D.3.1}$$

D is the rate of strain tensor (e.g. shear rate tensor). It is a tool to describe the stretching rate and angular rate of deformation of a volume element:

$$\mathbf{D} = \frac{1}{2}\left(\mathbf{L} \times \mathbf{L}^T\right) = \frac{1}{2}\begin{pmatrix} 2\frac{\partial u_r}{\partial r} & \left(\frac{\partial u_r}{\partial \Theta} + \frac{\partial u_\Theta}{\partial r}\right) & \left(\frac{\partial u_r}{\partial x} + \frac{\partial u_x}{\partial r}\right) \\ \left(\frac{\partial u_\Theta}{\partial r} + \frac{\partial u_r}{\partial \Theta}\right) & 2\frac{\partial u_\Theta}{\partial \Theta} & \left(\frac{\partial u_\Theta}{\partial x} + \frac{\partial u_x}{\partial \Theta}\right) \\ \left(\frac{\partial u_x}{\partial r} + \frac{\partial u_r}{\partial x}\right) & \left(\frac{\partial u_x}{\partial \Theta} + \frac{\partial u_\Theta}{\partial x}\right) & 2\frac{\partial u_x}{\partial x} \end{pmatrix} = \frac{1}{2}\begin{pmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{pmatrix} \tag{D.3.2}$$

Here **L** is the velocity gradient tensor:

$$\mathbf{L} = \nabla \vec{\mathbf{u}} = \begin{pmatrix} \frac{\partial u_r}{\partial r} & \frac{\partial u_r}{\partial \Theta} & \frac{\partial u_r}{\partial x} \\ \frac{\partial u_\Theta}{\partial r} & \frac{\partial u_\Theta}{\partial \Theta} & \frac{\partial u_\Theta}{\partial x} \\ \frac{\partial u_x}{\partial r} & \frac{\partial u_x}{\partial \Theta} & \frac{\partial u_x}{\partial x} \end{pmatrix} \tag{D.3.3}$$

It describes the change of each velocity component (u_r, u_Θ, u_x) in the spatial directions (r, Θ, x). With Eqn. (D.3.1) and Eqn. (D.3.2) the shear rate can be written as:

$$\dot{\gamma} = \sqrt{2 \cdot \left(D_{11}{}^2 + D_{22}{}^2 + D_{33}{}^2 + 2 \cdot D_{12}{}^2 + 2 \cdot D_{13}{}^2 + 2 \cdot D_{23}{}^2\right)} \tag{D.3.4}$$

$$\dot{\gamma} = \sqrt{2 \cdot \left(\left(\frac{\partial u_r}{\partial r}\right)^2 + \left(\frac{\partial u_\Theta}{\partial \Theta}\right)^2 + \left(\frac{\partial u_x}{\partial x}\right)^2 + \frac{1}{2}\left(\frac{\partial u_r}{\partial \Theta} + \frac{\partial u_\Theta}{\partial r}\right)^2 + \frac{1}{2}\left(\frac{\partial u_r}{\partial x} + \frac{\partial u_x}{\partial r}\right)^2 + \frac{1}{2}\left(\frac{\partial u_\Theta}{\partial x} + \frac{\partial u_x}{\partial \Theta}\right)^2\right)} \tag{D.3.5}$$

In a fully developed, symmetrical fluid flow following assumptions can be made:

$$\begin{aligned} &u_r = 0 \quad \rightarrow \frac{\partial u_r}{\partial r} = 0 \quad \frac{\partial u_r}{\partial \Theta} = 0 \quad \frac{\partial u_r}{\partial x} = 0 \\ &u_\Theta = 0 \quad \rightarrow \frac{\partial u_\Theta}{\partial r} = 0 \quad \frac{\partial u_\Theta}{\partial \Theta} = 0 \quad \frac{\partial u_\Theta}{\partial x} = 0 \\ &\text{fully developed flow} \rightarrow \frac{\partial u_x}{\partial x} = 0 \end{aligned} \tag{D.3.6}$$

This results in:

$$\dot{\gamma} = \sqrt{\left(\frac{\partial u_x}{\partial r}\right)^2 + \left(\frac{\partial u_x}{\partial \Theta}\right)^2} \tag{D.3.7}$$

With the velocity profile and its gradients [46]:

$$u_x = \left[1 - \frac{\eta}{4A_1} + \sum_{i=1}^{\infty} \frac{C_i}{A_1}\left(\eta^{mi} \cos(m\Theta)\right)\right] \cdot \mathrm{U}_{\max} \qquad \eta = \frac{\mathrm{r} \tan\left(\frac{\pi}{m}\right)}{s}$$

$$u_x = \left[1 - r^2 \left(\frac{\tan\left(\frac{\pi}{m}\right)}{s}\right)^2 \frac{1}{4A_1} + \sum_{i=1}^{\infty} \frac{C_i}{A_1} r^{mi} \left(\frac{\tan\left(\frac{\pi}{m}\right)}{s}\right)^{mi} \cos(m\Theta)\right] \cdot \mathrm{U}_{\max}$$

$$\frac{\partial u_x}{\partial r} = \left[-2r \left(\frac{\tan\left(\frac{\pi}{m}\right)}{s}\right)^2 \frac{1}{4A_1} + \sum_{i=1}^{\infty} \frac{C_i}{A_1} mi \cdot r^{(mi-1)} \left(\frac{\tan\left(\frac{\pi}{m}\right)}{s}\right)^{mi} \cos(m\Theta)\right] \cdot \mathrm{U}_{\max}$$

$$\frac{\partial u_x}{\partial \Theta} = \left[-\sum_{i=1}^{\infty} \frac{C_i}{A_1} r^{mi} \left(\frac{\tan\left(\frac{\pi}{m}\right)}{s}\right)^{mi} \mathrm{m} \cdot \sin(m\Theta)\right] \cdot \mathrm{U}_{\max} \qquad \text{(D.3.8)}$$

D.4 Half Orbit Time and Traveling Length

With Eqn. (D.3.7) it is now possible to determine the time and traveling length of the fibre closest to the wall by applying the Jeffery orbit.

$$t^+ = 2\pi \frac{AR + \frac{1}{AR}}{\dot{\gamma}} \qquad \text{(D.4.1)}$$

$$t_{0.5orbit}{}^+ = t^+/2 \qquad \text{(D.4.2)}$$

$$s_{0.5orbit}{}^+ = u_x \; t_{0.5orbit}{}^+ \qquad \text{(D.4.3)}$$

Note, the shear rate is higher at the wall and smaller towards the center of the channel. The higher the aspect ratio, the longer a fibre - time and length wise - needs to turn.

D.5 Fibre Orbit: Analytical vs. Numerical Results

The analytical Jeffery orbit [45] results and the numerical results for the dimensionless time $t_{0.5orbit}{}^+$ and length $s_{0.5orbit}{}^+$ units for a half orbit are compared.

The analytical results determined by applying the Jeffery orbit can be seen in Table 20.

Table 20: Jeffery orbit calculation.

class	d_{Major}	AR	r_{max}^{+}	u_x	$\dot{\gamma}$	$t_{0.5orbit}^{+}$	$s_{0.5orbit}^{+}$
1	0.04	2	0.48	0.18	9.14	0.86	0.15
2	0.10	5	0.45	0.44	8.29	1.97	0.86
3	0.20	10	0.40	0.82	6.99	4.45	3.72
4	0.30	15	0.35	1.14	5.83	8.12	9.24
5	0.40	20	0.30	1.40	4.79	13.15	**18.45**

For the numerical simulation a straight channel with a laminar velocity profile (*Re*=500) is used (see Fig. 61). Two simulations, one with a fibre from the shortest fibre class d_{Major}=0.04. starting at r_{max}^{+}=0.48 and *Θ=0,* oriented in streamwise direction and one with a fibre from the longest fibre class d_{Major}*=0.40,* starting at r_{max}^{+}*=0.30* and *Θ=0,* oriented in streamwise direction. The rotational energy of the system was monitored. After each 180° spin a local minimum of the rotational energy occurs. The $t_{0.5orbit}^{+}$ at which this minimum is reached can then be compared to the value form the Jeffery orbit calculations. The difference between the numerical and the analytical results are small, Δt^{+}_{Diff}= 4.5% for the shortest and Δt^{+}_{Diff}=1.4% for the longest fibre. Since the longest fibre is crucial for the inlet length dimensioning the considered error is 1.4%.

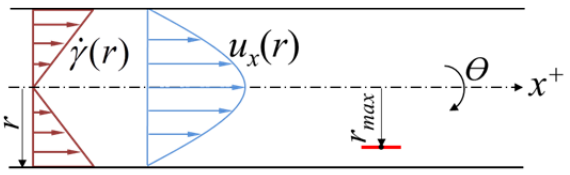
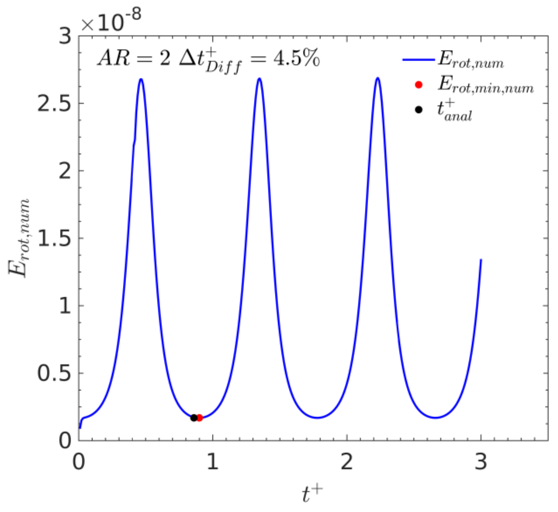


Fig. 61: Velocity and shear rate profile in channel with laminar flow field, fibre at r_{max}.



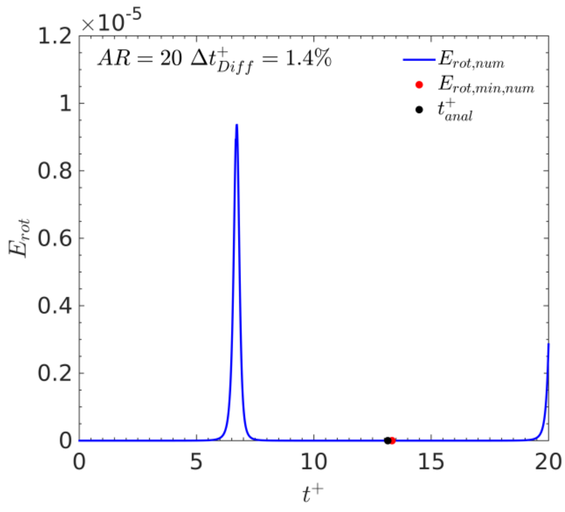


Fig. 62: Comparison of the numerical and the analytical solutions of a 180° spin of a fibre in a linear flow field. Blue curve: Rotational energy in the system. Red dot: first minima after 180° spin. Black dot: analytical dimensionless time result for a 180° spin. Left panel: shortest fibre *AR*=2. Right panel: longest fibre *AR*=20.

Appendix E Dimensionless Numbers

Fluid Reynolds number

$$Re = \frac{\text{inertial forces}}{\text{viscose forces}} = \frac{u\,L}{\nu} = \frac{\rho\,u\,L}{\mu} \qquad \mu = \nu\rho \tag{E.1.1}$$

u volume-averaged fluid velocity, L characteristic length, ρ fluid density, μ dynamic viscosity, ν kinematic viscosity of the fluid [14].

Particle Reynolds number

$$Re_p = \frac{\text{inertial forces}}{\text{viscose forces}} = \frac{d_p u_{rel}}{\nu} \tag{E.1.2}$$

d_p particle diameter, u_{rel} relative velocity (i.e., fluid minus particle velocity), ν kinematic viscosity of the fluid [31].

Stokes number

$$St = \frac{\tau\,u}{L} \qquad \tau = \frac{\rho_p\,d_p^{\,2}}{18\mu} \tag{E.1.3}$$

τ Stokes relaxation time, u volume-averaged fluid velocity, L characteristic length, ρ_p particle density, d_p particle diameter, μ dynamic viscosity of the fluid [67].

Courant number

$$C = \frac{\Delta t\ u}{\Delta x} \tag{E.1.4}$$

The Courant number is fundamental for numerical simulations. Simulation with C $\leq$ 1 are supposed to be stable. The Courant number indicates the distance a characteristic dimension moves within a single time step [20].

Appendix F MATLAB Code

Appendix F is only available in the electronic version.

CFD

- time-average data over outlet patches
- volumetric flow fraction and pressure loss

LIGGGHTS

- fibre-fibre test case pre and post processing
- fibre flock generation and stabilization pre and post processing

CFDEM

- straight channel
  - fibre-wall collision impact study pre and post processing
  - rotation study numerical versus Jeffery pre and post processing
- T-Channel alias T-Junction
  - critical separation line pre and post processing
  - downwards movement at junction pre and post processing
  - fiber orientation and separation efficiency pre and post processing

Repeatedly Used Functions

- formatting_GNU
- formatting_MATLAB
- readdump_all
- ShearRateSCS
- velSCS

```
% CFD_timeAveragedVolumeFraction
% Program to determine Volume Fraction in T-Channel
% (c) Lisa Koenig, August 2015, TU Graz
%
% phi  ... volume fraction = outJ_UxTimeAveraged/outC_UzTimeAveraged
% outJ ... outlet side channel, main stream direction!
% outC ... outlet main channel, main stream direction!
clear all; close all; clc

%%  -------------------------------- User Input  ----------------------- %
rho=1; % fluid density

DirInC.U  = '../../postProcessing/inC_U/330/';   % Source Directory
FileInC.U = 'faceSource.dat';                    % Source File
DirInC.p  = '../../postProcessing/inC_p/330/';   % Source Directory
FileInC.p = 'faceSource.dat';                    % Source File

DirOutJ.U  = '../../postProcessing/outJ_U/330/';  % Source Directory
FileOutJ.U = 'faceSource.dat';                    % Source File

DirOutC.U  = '../../postProcessing/outC_U/330/'; % Source Directory
FileOutC.U = 'faceSource.dat';                   % Source File
pOutC=0.0;                                       % static pressure/rho

outDir = '.';                                    % Output Directory
outputFile ='volFrac_pLoss_timeAveraged.txt';    % Output File

%%  --------------------------- Read Data from Files  ------------------ %
% outC
DataOutC=func_averageAreaNormalIntegrateOverPatch_U(DirOutC.U,FileOutC.U);

% outJ
DataOutJ=func_averageAreaNormalIntegrateOverPatch_U(DirOutJ.U,FileOutJ.U);

% inC
DataInC=func_averageAreaNormalIntegrateOverPatch_U(DirInC.U,FileInC.U);
DataInC=func_average_p(DirInC.p,FileInC.p,DataInC);

%%  ---------------------- Determine Volume Fraction Phi --------------- %
% Volume Fraction phi
phi=DataOutJ.U1TimeAveraged*0.5 / DataInC.U1TimeAveraged

%% ----------------------- Determine Pressure Loss --------------------- %
% inC to outC

% static pressure/rho
pStatInC  = DataInC.pTimeAveraged
pStatOutC = pOutC
% dynamic Pressure/rho
pDynInC  = DataInC.U1TimeAveraged^2/2;
pDynOutC = DataOutC.U1TimeAveraged^2/2;

% Pressure loss / rho
pLoss = pStatInC - pStatOutC + pDynInC - pDynOutC


%%  ---------------------------- Write Putput File --------------------- %
% Output in .txt file
outFile=fopen(fullfile(outDir,outputFile),'w');
fprintf(outFile,'\n %s \n'    , 'time-averaged U in patch inC');
fprintf(outFile,'%10.8f %s \n', DataInC.U1TimeAveraged, ' U1Averaged ');
fprintf(outFile,'%10.8f %s \n', DataInC.U2TimeAveraged, ' U2Averaged ');
fprintf(outFile,'%10.8f %s \n', DataInC.U3TimeAveraged, ' U3Averaged ');

fprintf(outFile,'\n %s \n'    , 'time-averaged U in patch outJ');
fprintf(outFile,'%10.8f %s \n', DataOutJ.U1TimeAveraged, ' U1Averaged ');
fprintf(outFile,'%10.8f %s \n', DataOutJ.U2TimeAveraged, ' U2Averaged ');
fprintf(outFile,'%10.8f %s \n', DataOutJ.U3TimeAveraged, ' U3Averaged ');

fprintf(outFile,'\n %s \n'    , 'time-averaged U in patch outC');
fprintf(outFile,'%10.8f %s \n', DataOutC.U1TimeAveraged, ' U1Averaged ');
fprintf(outFile,'%10.8f %s \n', DataOutC.U2TimeAveraged, ' U2Averaged ');
fprintf(outFile,'%10.8f %s \n', DataOutC.U3TimeAveraged, ' U3Averaged ');

fprintf(outFile,'\n %s \n'    , 'time-averaged p in patch inC');
fprintf(outFile,'%10.8f %s \n', DataInC.pTimeAveraged, ' pAveraged ');

fprintf(outFile,'\n %s \n'    ,...
       'abs(phi): volume fraction outJ_U1TimeAveraged/inC_U1TimeAveraged');
fprintf(outFile,'%10.8f \n', abs(phi));

fprintf(outFile,'\n %s \n'    ,...
       'pLoss/rho: averaged pressure loss p/rho');
fprintf(outFile,'%10.8f \n', pLoss);

%%  ---------------------------- End of Program ------------------------ %
disp('End of program.')
```

```matlab
function Data=func_average_p(sourceDir,sourceFile,Data)

% Function to read and average the area average p values over time
% over patch
% (c) Lisa Koenig, Aug 2015, TU GRAZ
%
% Read file: sourceFile  (e.g. sourceDir  = './inX/200/';)
% in directory: sourceDir (e.g. sourceFile = 'inX_U.txt';)
%
% needed File format:
% # Source : patch X
% # Faces  : 900
% # Time  sum(magSf)      areaAverage(p)
% 330.0119    1           1.432586
% ...

% Open Source file
inFile=fopen(fullfile(sourceDir,sourceFile),'r');

% Count Lines of File
  nLines = 0;
  tline = fgetl(inFile);
  while ischar(tline)
     tline = fgetl(inFile);
     nLines = nLines+1;
  end
  frewind(inFile); % go back to first line

% Read Source File
nHeaders = 3;           % number of headerlines
M  = textscan(inFile, '%f%f%f', ...
                      nLines-nHeaders, ...
                      'headerLines', nHeaders);
frewind(inFile); % go back to first line
%celldisp(M);     % display Cell Array

Data.Time=[M{1,1}];
Data.Area=[M{1,2}];
Data.p=[M{1,3}];

% Determine time-averaged p
Data.nValues=length(Data.p);
Data.pTimeAveraged=sum(Data.p)/length(Data.p);
end
```

```matlab
function Data=func_averageAreaNormalIntegrateOverPatch_U(sourceDir,sourceFile)

% Function to read and average the area normal Integrate Value U
% over patch
% (c) Lisa Koenig, Aug 2015, TU GRAZ
%
% Read file: sourceFile  (e.g. sourceDir  = './inX/200/';)
% in directory: sourceDir (e.g. sourceFile = 'inX_U.txt';)
%
% needed File format:
% # Source : patch X
% # Faces  : 900
% # Time  sum(magSf)      areaNormalIntegrate(U)
% 0.00059952038   1       (-1 0 0)
% 0.0013180822    1       (-1 0 0)
% 0.0021793636    1       (-1 0 0)
% 0.0032121865    1       (-1 0 0)
% ...

% Open Source file
inFile=fopen(fullfile(sourceDir,sourceFile),'r');

% Count Lines of File
  nLines = 0;
  tline = fgetl(inFile);
  while ischar(tline)
     tline = fgetl(inFile);
     nLines = nLines+1;
  end
  frewind(inFile); % go back to first line

% Read Source File
nHeaders = 3;           % number of headerlines
M  = textscan(inFile, '%f%f(%f%f%f)', ...
                      nLines-nHeaders, ...
                      'headerLines', nHeaders);
frewind(inFile); % go back to first line
%celldisp(M);     % display Cell Array

Data.Time=[M{1,1}];
Data.Area=[M{1,2}];
Data.U1=[M{1,3}];
Data.U2=[M{1,4}];
Data.U3=[M{1,5}];

% Determine time-averaged U (Ux,Uy,Uz) values
Data.nValues=length(Data.U1);
Data.U1TimeAveraged=sum(Data.U1)/length(Data.U1);
Data.U2TimeAveraged=sum(Data.U2)/length(Data.U2);
Data.U3TimeAveraged=sum(Data.U3)/length(Data.U3);
end
```

```
% Programm to Plot Data
% (c) Lisa Koenig
%
% INFO: Merge Data by hand! Sort!
% SCW ... side channel width
%
clear all; close all; clc

global homeDir inDir resDir FigVisible

%% Directories
homeDir=pwd;
inDir='./';
resDir='./';
FigVisible='on';
printAs='-dpng';

inFileName='VolumeFlowFraction_*.txt';

%% Get input files
cd (inDir)
inFiles=dir(inFileName);
cd(homeDir)
nFiles=length(inFiles);

%% Read Input Files
for i=1:1:nFiles
    % Open input File
    fopen(fullfile(inDir,inFiles(i).name),'r');

    % Read Data from File
    File(i).Data=dlmread(fullfile(inDir,inFiles(i).name),'',1,0);
    File(i).nDataPts=size(File(i).Data,1);
end

%% ------------------ Create Figure Volume Fraction phi+ ----------------
% run formating
run formatting.m;
rect=[0 10 600 500];

% Get Screen Size
scrsz = get(0,'ScreenSize');
Name='VolumeFlowFraction_Re500_Phi_dp';
fig1=figure('Name',Name,...
    'Position',rect,...
    'Visible',FigVisible); % [left, bottom, width, height]

% Inlet Length 20
Linlet=20;
for i=1:1:nFiles
   x=File(i).Data(:,2);     % dp
   y=File(i).Data(:,5);     % phi+
   Re=File(i).Data(1,1);    % Re
   angle=File(i).Data(1,3); % side channel angle
   SCW=File(i).Data(1,4);   % SCW
   plot(x,y,...
    symbolArray{i},...
    'LineStyle','-',...
    'MarkerSize', markerSize*1.1,...
    'MarkerFaceColor', faceColorArray{i}); hold on;
    legendTextPhi20{i}=['$ L_{inlet}^{+}=',num2str(Linlet),',$ ',...
                        '$ SCW^{+}=',num2str(SCW,'%2.1f'),',$ ',...
                        '$ \alpha =',num2str(angle),'$'];


end

% Inlet Length 40
Linlet=40;
for i=1:1:nFiles
   x=File(i).Data(:,2);     % dp
   y=File(i).Data(:,7);     % phi+
   Re=File(i).Data(1,1);    % Re
   angle=File(i).Data(1,3); % side channel angle
   SCW=File(i).Data(1,4);   % SCW
   plot(x,y,...
    symbolArray{i+4},...
    'LineStyle','-',...
    'MarkerSize', markerSize*.7,...
    'MarkerFaceColor', faceColorArray{i+4}); hold on;
    legendTextPhi40{i}=['$ L_{inlet}^{+}=',num2str(Linlet),',$ ',...
                        '$ SCW^{+}=',num2str(SCW,'%2.1f'),',$ ',...
                        '$ \alpha =',num2str(angle),'$'];
end
legendTextPhi=[legendTextPhi20 legendTextPhi40];

legend(legendTextPhi,'interpreter','latex')
xlabel('$ \Delta p^{+}$','interpreter','latex')
ylabel('$ \Phi ^{+}$','interpreter','latex')

xlim([-0.3 0.4])
ylim([0.0 0.4])
set(gca,'XMinorTick','on','YMinorTick','on');
set(gca,'XTick',[-0.3:0.1:0.4]);
set(gca,'YTick',[0:0.1:.4]);
grid off
box on
legend boxoff
set(0,'defaultaxesfontsize',fontSizeAxis);
set(0,'defaulttextfontsize',fontSizeAxis);
set(gca,'FontSize',fontSizeAxis);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',fontSizeLabel); set(ylhand,'fontsize',fontSizeLabel)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

% Save Figure
print(printAs,'-r300', fullfile(resDir,Name));


%% ------------------ Create Figure Pressure Loss/rho ----------------
% Pressure Loss with inlet length 20 and inlet length 40

% Get Screen Size
scrsz = get(0,'ScreenSize');
Name='pLossRho_Inlet20_Re500_dp';
fig2=figure('Name',Name,...
    'Position',rect,...
    'Visible',FigVisible); % [left, bottom, width, height]

% Inlet Length 20
Linlet=20;
for i=1:1:nFiles
   x=File(i).Data(:,2);      % dp+
   y=File(i).Data(:,6);      % phiPlus
   Re=File(i).Data(1,1);     % Re
   angle=File(i).Data(1,3);  % side channel angle
   SCW=File(i).Data(1,4);    % SCW
   plot(x,y,...
    symbolArray{i},...
    'LineStyle','-',...
    'MarkerSize', markerSize*1.1,...
    'MarkerFaceColor', faceColorArray{i}); hold on;
    legendText20PLoss{i}=['$ SCW^{+}= ',num2str(SCW,'%2.1f'),',$ ',...
                          '$ \alpha =',num2str(angle),'$'];
end
```

```

legend(legendText20PLoss,'interpreter','latex',...
     'location','Northwest')
xlabel('$ \Delta p^{+}$','interpreter','latex')
ylabel('$ p_{loss}^{+} / \rho^{+}$','interpreter','latex')

xlim([-0.3 0.41]);
ylim([1.2 1.6]);
set(gca,'XMinorTick','on','YMinorTick','on');
set(gca,'XTick',[-0.3:0.1:0.4]);
set(gca,'YTick',[1.2:0.1:1.6]);
box on
legend boxoff
set(0,'defaultaxesfontsize',fontSizeAxis);
set(0,'defaulttextfontsize',fontSizeAxis);
set(gca,'FontSize',fontSizeAxis);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',fontSizeLabel); set(ylhand,'fontsize',fontSizeLabel)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

% Save Figure
print(printAs,'-r300', fullfile(resDir,Name));

% ---------------------------------------------------------------------- %

% Get Screen Size
scrsz = get(0,'ScreenSize');
Name='pLossRho_Inlet40_Re500_dp';
fig3=figure('Name',Name,...
    'Position',rect,...
    'Visible',FigVisible); % [left, bottom, width, height]

% Inlet Length 40
Linlet=40;
for i=1:1:nFiles
   x=File(i).Data(:,2);
   y=File(i).Data(:,8);
   Re=File(i).Data(1,1);
   angle=File(i).Data(1,3)
   SCW=File(i).Data(1,4);
   plot(x,y,...
    symbolArray{i},...
    'LineStyle','-',...
    'MarkerSize', markerSize,...
    'MarkerFaceColor', faceColorArray{i}); hold on;
    legendText40PLoss{i}=['$ SCW^{+}=',num2str(SCW,'%2.1f'),',$ ',...
                          '$ \alpha =',num2str(angle),'$'];
end

legend(legendText40PLoss,'interpreter','latex',...
     'location','Northwest')
xlabel('$ \Delta p^{+}$','interpreter','latex')
ylabel('$ p_{loss}^{+} / \rho^{+}$','interpreter','latex')

xlim([-0.1 0.4])
ylim([2.4 2.8])
set(gca,'XMinorTick','on','YMinorTick','on')
set(gca,'XTick',[-0.1:0.1:0.4])
set(gca,'YTick',[2.4:0.1:2.8])
grid off
box on
legend boxoff
set(0,'defaultaxesfontsize',fontSizeAxis);
set(0,'defaulttextfontsize',fontSizeAxis);
set(gca,'FontSize',fontSizeAxis);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',fontSizeLabel); set(ylhand,'fontsize',fontSizeLabel)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

% Save Figure
print(printAs,'-r300', fullfile(resDir,Name));

disp('End of program');
```

```matlab
% This programm contains the preprocessing calculation
% (c) L. Koenig, 2015
%
% 1) Input data is read
% 2) pre-collision data is determined (position of fibre B acc. to CollisionAll)
% 3) post-collision data is determined (velocity and angular velocity)
%    using an analytical solution
% 4) Enegry and linear Momentum conservation is checked for analytical
%    solution
% 5) Analytical Results are saved in a .txt-file
% 6) LIGGGHTS input files are prepared
%    - Fibre properties that stay the same in all cases
%    - Fibre properties that vary in all cases
clear all; close all; clc;

%% Read Input data:
inStat=0;
[Fib,CollisionAll,alpha]=func_inputData(inStat);

%% Precalculation
% Calc Mass of fibres
Fib.mass=Fib.rho*Fib.dMinor*Fib.dMinor*Fib.dMajor*pi/6;%=Fib.massB=Fib.massA;

% Calc Moment of Inertia of fibres
% Inertia around y Iy= 1/5 mass ((dMajor/2)^2 + (dMinor/2)^2)
Fib.IA=1/5*Fib.mass*((Fib.dMajor/2)^2+(Fib.dMinor/2)^2);
Fib.IB=Fib.IA;

% Calculate Coefficient of Restitution
CollisionAll.CoR = func_CalcCoR(Fib.mass, Fib.stiffnessNormal,Fib.dampingNormal);

%% Analytical Solution
CollisionAll=func_Analytic(alpha,Fib,CollisionAll);

%% Prepare LIGGGHTS(R) input files for numerical solution
curDir=pwd;
cd ..
cd ..
MainDir=pwd;
cd (curDir)

% Fibre Properties, constant for all cases
fileName='FibDataConst.txt';
FibDataConst = fopen(fullfile(MainDir,'preProcessing',fileName),'w');
fprintf(FibDataConst,'%s %6.4f %s \n',  'variable dMajor             equal  ',Fib.dMajor,            ' # major diameter of ellipsoid (=total length of spherocylinder)');
fprintf(FibDataConst,'%s %6.4f %s \n',  'variable dMinor             equal  ',Fib.dMinor,            ' # minor diameter of ellipsoid');
fprintf(FibDataConst,'%s %6.4f %s \n',  'variable rhoParticle        equal  ',Fib.rho,               ' # particle/fibre density');
fprintf(FibDataConst,'%s %6.4f %s \n',  'variable stiffnessNormal    equal  ',Fib.stiffnessNormal,   ' # is taken here  for the spring constant for the normal contact');
fprintf(FibDataConst,'%s %6.4f %s \n',  'variable dampingNormal      equal  ',Fib.dampingNormal,     ' # small damping; 5e-2, normal direction to gently remove kinetic energy; coefficient of restitution = 1 sets the damping to 0');
fprintf(FibDataConst,'%s %6.4f %s \n',  'variable stiffnessTang      equal  ',Fib.stiffnessTang,     ' # set to zero');
fprintf(FibDataConst,'%s %6.4f %s \n',  'variable dampingTang        equal  ',Fib.dampingTang,       ' # set to zero');
fprintf(FibDataConst,'%s %12.10f %s \n','variable roughFact          equal  ',Fib.roughFact,         ' # fraction of dMinor giving the particle roughness (will model fibrils on fibre), can not be zero!');
fprintf(FibDataConst,'%s %6.4f %s \n',  'variable roughStiffFact     equal  ',Fib.roughStiffFact,    ' # fraction of the fibre normal stiffness used to calculate the repelling force due to roughness contact');
fprintf(FibDataConst,'%s %6.4f %s \n',  'variable frictionCoeff      equal  ',Fib.frictionCoeff,     ' # set to zero');
fprintf(FibDataConst,'%s %6.4f %s \n',  'variable liquidDynViscosity equal  ',Fib.liquidDynViscosity,' # lubrication model for fibre/fibre interaction if zero -> no lubr.');
fclose(FibDataConst);

% Fibre Properties, vary for all cases
for i=1:1:length(alpha);
    fileName=['FibData_Case',num2str(i),'.txt'];
FibData=fopen(fullfile(MainDir,'preProcessing',fileName),'w');
fprintf(FibData,'%s %1i %s %6.4f \n',...
        '# Case_',i,' alpha = ',alpha(i)/pi*180);
fprintf(FibData,'%s %6.4f \n','variable rotAngle equal ',0.0);
fprintf(FibData,'%s %6.4f \n','variable initialPosX equal ',0.0);
fprintf(FibData,'%s %6.4f \n','variable initialPosY equal ',0.0);
fprintf(FibData,'%s %6.4f \n','variable initialPosZ equal ',0.0);
fprintf(FibData,'%s %6.4f \n','variable velPartInitX equal ',CollisionAll.vA1(1));
fprintf(FibData,'%s %6.4f \n','variable velPartInitY equal ',CollisionAll.vA1(2));
fprintf(FibData,'%s %6.4f \n','variable velPartInitY equal ',CollisionAll.vA1(3));
fprintf(FibData,'%s \n','');
fprintf(FibData,'%s %6.4f \n','variable rotAngle2 equal ',180-(90+alpha(i)/pi*180));
fprintf(FibData,'%s %6.4f \n','variable initialPosX_2 equal ',CollisionAll.cmB1(i,1));
fprintf(FibData,'%s %6.4f \n','variable initialPosY_2 equal ',CollisionAll.cmB1(i,2));
fprintf(FibData,'%s %6.4f \n','variable initialPosZ_2 equal ',CollisionAll.cmB1(i,3));
fprintf(FibData,'%s %6.4f \n','variable velPartInitX_2 equal ',CollisionAll.vB1(1));
fprintf(FibData,'%s %6.4f \n','variable velPartInitY_2 equal ',CollisionAll.vB1(2));
fprintf(FibData,'%s %6.4f \n','variable velPartInitZ_2 equal ',CollisionAll.vB1(3));
end

disp('preProcessing done...');
disp('Results from analytical Solution can be found in:');
disp('./postProcessing/AnalyticResults.txt');
disp('and');
disp('LIGGGHTS input data can be found in:');
disp('./preProcessing/');
```

```
function [Fib,CollisionAll,alpha]=func_inputData(inStat)

%% Input data
% Input can be changes by USER
%Fibre Data
Fib.dMajor=0.40;     % Major diameter of fibres
Fib.dMinor=0.02;     % Minor diameter of fibre
Fib.rho=1.27;        % density of fibre A and fibre B

% Fibre Properties
Fib.stiffnessNormal   = 2e4;      % is taken here  for the spring constant for the normal contact
Fib.dampingNormal      = 1e-2;     % small damping (5e-2), normal direction to gently remove kinetic energy; coefficient of restitution = 1↙
sets the damping to 0
Fib.stiffnessTang      = 0;        % set to zero
Fib.dampingTang        = 0;        % set to zero
Fib.roughFact          = 1e-2;     % fraction of dMinor giving the particle roughness (will model fibrils on fibre), can not be zero!
Fib.roughStiffFact     = 1e-3;     % fraction of the fibre normal stiffness used to calculate the repelling force due to roughness contact 1e-2
Fib.frictionCoeff      = 0;        % set to zero
Fib.liquidDynViscosity= 0;        % lubrication model for fibre/fibre interaction if zero -> no lubr.

% Velocity and Angular Velocity
CollisionAll.vA1 =[0 0 0];          % initial (pre-collision) fibre velocity, A
CollisionAll.vB1 =[-10 0 0];        % initial (pre-collision) fibre velocity, B
CollisionAll.omA1=[0 0 0];          % initial (pre-collision) fibre angular velocity, A
CollisionAll.omB1=[0 0 0];          % initial (pre-collision) fibre angular velocity, B

% Distance factor to the define the collsion point on fibre B
CollisionAll.DistFact=0.6;

% Alpha - Angle between  fibre center lines
alpha=-[0 15 30 45 60 75 90]/180*pi;   %  alpha=-[0 20 40 60 80]/180*pi;

inStat=1;
end
```

```
function CoR=func_CalcCoR(mass, springConst, dashConst)
% Calculation of the Coefficient of Restitution (CoR)
% Source: Schwager, PÃ¶schl (2008) - Coefficient of restitution and linear
% dashpot model revisited
% Eqn. (10), tc0 = pi/omega, Eqn (21)
%
% Input:
% mass        ... mass of particle
% springConst ... spring constant of spring/dashpot model
% dashConst   ... dashpot constant of spring/dashpot model

om0=sqrt(2*springConst/mass);
beta=dashConst/mass;
om=sqrt(om0^2-beta^2);
Com = sqrt(beta^2-om0^2);

if beta < om0/(2)^.05
    CoR = exp(-beta/om *(pi-atan(2*beta*om/(om^2-beta^2))));
elseif beta >= om0/(2)^.05 && beta < om0
    CoR = exp(-beta/om * atan(2*beta*om/(om^2-beta^2)));
elseif beta > om0
    CoR = exp(-beta/Com * log((beta+Com)/(beta-Com)));
end

end
```

```matlab
function CollisionAll=func_Analytic(alpha,Fib,CollisionAll)

% Calculation of the Analytical Solution

% 1) Collision Point(s)
% 2) Final (post-collision) velocity and angular velocity
% 3) Inital (pre-collision) and final (post-collision) total energy
% 4) Save Data in .txt -File

% Redefine variable
dMajor=Fib.dMajor;
dMinor=Fib.dMinor;

%% 1) Collision Point(s)
% ==========================
% Calculation of the collision points and the necessary starting position
% for fibre B
% Note: Only 2D on xz-plane, rotation around y-axis

% Collision point and vectors for first collision (alpha=0)
% -----------------------------------------------------------

DistFact=CollisionAll.DistFact;

% Define initial possitions with alpha=0, fibres are normal to each other
% Fibre A
Start.cmA1  =[0 0 0];                        % center of mass of fibre A, from origin
Start.M_A1  =[(dMajor-dMinor)/2 0 0]; % center of capsule A right half sphere, from origin

% 1st collision point, alpha=0
Start.P1  =[dMajor/2 0 0];               % point of 1st collision, from origin

% Fibre B
Start.cmB1 =...
    [(dMajor+dMinor)/2 0 -(dMajor-dMinor)/2*DistFact]; % center of mass of fibre B, from origin; x (=DistFact) is just a factor

% Vector calculation
Start.M_A1_P1 = Start.P1 - Start.M_A1;   % Vec: center of capsule A right half sphere to collision point 1
Start.cmA1_P1 =Start.M_A1+Start.M_A1_P1; % Vec: center of mass (A) to collision point 1
Start.cmB1_P1 =Start.cmA1_P1-Start.cmB1; % Vec: center of mass (B) to collision point 1
Start.M_A1_cmB1=Start.cmB1 - Start.M_A1; % Vec: center of capsule A right half sphere to center of mass (B1)

% Normal Vector for collision /Tangent on half sphere in P1
Start.nVec1=Start.M_A1_P1 / ...
    sqrt(Start.M_A1_P1(1)^2 + Start.M_A1_P1(2)^2 +...
    Start.M_A1_P1(3)^2); % Collision normal unit vector

% Collision point and vectors for all collisions (all alpha)
% -----------------------------------------------------------
% Fibre A does not change position, only Fibre B
for i=1:1:length(alpha)
Rrot=[cos(alpha(i)) 0 -sin(alpha(i));
      0          1     0     ;
      sin(alpha(i)) 0 cos(alpha(i))];                                  %rotation Matrix around y-axis (xz-plane)

CollisionAll.alphaRad(i,1)=alpha(i);
CollisionAll.alphaDeg(i,1)=alpha(i)/pi*180;

CollisionAll.M_A1_cmB1(i,:)= (Rrot*Start.M_A1_cmB1.').';              % Vec: center of capsule (A) right half sphere to center of mass (B2)
CollisionAll.cmB1(i,:)      = CollisionAll.M_A1_cmB1(i,:) + Start.M_A1;% center of mass of fibre B2, from origin
CollisionAll.M_A1_Pc(i,:)  = (Rrot*Start.M_A1_P1.').';                % Vec: center of capsule A right half sphere to collision point 4
CollisionAll.Pc(i,:)=Start.M_A1+CollisionAll.M_A1_Pc(i,:);
CollisionAll.cmA1_Pc(i,:)  = Start.M_A1 + CollisionAll.M_A1_Pc(i,:);  % Vec: Center of Mass (A) to collision Point
CollisionAll.cmB1_Pc(i,:)  = CollisionAll.M_A1_Pc(i,:)-...
                           CollisionAll.M_A1_cmB1(i,:);                  % Vec: Center of Mass (B) to collision Point
CollisionAll.nVec(i,:)      = -CollisionAll.M_A1_Pc(i,:) /...
  sqrt(CollisionAll.M_A1_Pc(i,1)^2 + CollisionAll.M_A1_Pc(i,2)^2 +...
                                CollisionAll.M_A1_Pc(i,3)^2);         % Collision normal unit vector
end


%% 2) Final (post-collision) velocity and angular velocity

CoR=CollisionAll.CoR;
mA=Fib.mass;
mB=Fib.mass;
IA=Fib.IA;
IB=Fib.IB;
omA1=CollisionAll.omA1;
omB1=CollisionAll.omB1;
vA1=CollisionAll.vA1;
vB1=CollisionAll.vB1;

for i=1:1:length(alpha)
% Initial (pre-collision) velocity of collsision point(s) at A and B
% Collision point from center of Fib.mass

rAP = CollisionAll.cmA1_Pc(i,:);
rBP = CollisionAll.cmB1_Pc(i,:);
nVec=CollisionAll.nVec(i,:);

% Calc
vAP1=vA1+cross(omA1,rAP);
VBP1=vB1+cross(omB1,rAP);

F_Val(i)= (dot((1+CoR).*(vAP1-VBP1),nVec))/...
(1/mA + 1/mB + ...
  dot( (cross(cross(rAP,nVec),rAP./IA) + ...
        cross(cross(rBP,nVec),rBP./IB))  ...
  ,nVec) ...
);

dvA(i,:)=-F_Val(i)/mA.*nVec;
dvB(i,:)= F_Val(i)/mB.*nVec;

domA(i,:)= -cross(rAP,(F_Val(i)/IA).*nVec);
domB(i,:)=  cross(rBP,(F_Val(i)/IB).*nVec);

CollisionAll.vA2(i,:)=vA1+dvA(i,:);
CollisionAll.vB2(i,:)=vB1+dvB(i,:);

CollisionAll.omA2(i,:)=omA1+domA(i,:);
CollisionAll.omB2(i,:)=omB1+domB(i,:);

end

%% 3) Inital (pre-collision) and final (post-collision) total energy and linear momentum
    vA2=CollisionAll.vA2;
    vB2=CollisionAll.vB2;
    omA2=CollisionAll.omA2;
    omB2=CollisionAll.omB2;

for i=1:1:length(alpha)
    % If there is no loss of energy to friction (damping = 0) or during
    % collisions (elasticity = 1) then the energy of the system should
    % not change.
    % no potential Energy; kinetic Energy only

    % kinetic Energy
    % translation Energy: ETrans = 1/2 m v^2
    ETransA1(i,:)=mA./2.*dot(vA1,vA1);
    ETransB1(i,:)=mB./2.*dot(vB1,vB1);
    CollisionAll.ETrans1(i,:)=ETransA1(i,:)+ETransB1(i,:);

    ETransA2(i,:)=mA./2.*dot(vA2(i,:),vA2(i,:));
    ETransB2(i,:)=mB./2.*dot(vB2(i,:),vB2(i,:));
    CollisionAll.ETrans2(i,:)=ETransA2(i,:)+ETransB2(i,:);

    % rotational Energy ERot = 1/2 I om^2
    ERotA1(i,:)=1/2*IA.*dot(omA1,omA1);
    ERotB1(i,:)=1/2*IB.*dot(omB1,omB1);
```

```
% This programm Plot the analyrtical and numerical results
% (c) L. Koenig, Stefan Radl, TU Graz
clear all; close all;clc

% gobal
global showFig resultDir
curDir=pwd;

%% General settings
interpreterName = 'latex';  % tex or latex; change: formatting.m / formating_MATLAB.m manually
showFig='off';              % show plots as figures
resultDir='./postProcessing/results/';  % folder where the plots are saved

% * * * * * * * * * * * * * * * * * * * * * * * * * * *
% END USER INPUT
% * * * * * * * * * * * * * * * * * * * * * * * * * * *
cd ..
cd ..
cd ./postProcessing
if exist('results') == 0 % if folder does not exist create it!
   mkdir('results')
end
%% Read data from analytical and from numerical (LIGGGHHTS) solution

% Analytical Results
% ===================
% (1)alpha (2)vA2x (3)vA2y (4)vA2z (5)VB2x (6)VB2y (7)VB2z (8)omA2x
% (9)omA2y (10)omA2z (11)omB2x (12)omB2y (13)omB2z (14)EKin1 (15)EKin2 (16)P1 (17)P2 (18)CoR
Analytic= dlmread('AnalyticResults.txt','',1,0); % skip 1 rows and 0 columns
CoR=Analytic(1,18)

% Numerical Results
% ===================
% (1)nCase (2)vAx1 (3)vAy1 (4)vAz1 (5)vAx2 (6)vAy2 (7)vAz2
% (8)omAx1 (9)omAy1 (10)omAz1 (11)omAx2 (12)omAy2 (13)omAz2 (14)vBx1
% (15)vBy1 (16)vBz1 (17)vBx2 (18)vBy2 (19)vBz2 (20)omBx1 (21)omBy1
% (22)omBz1 (23)omBx2 (24)omBy2 (25)omBz2 (26)Ekin1 (27)Epot1 (28)Etot1
% (29)Ekin2 (30)Epot2 (31)Etot2 (32)LinMomTot1 (33)AngMomTot1
% (34)LinMomTot2 (35)AngMomTot2
Numeric= dlmread('NumericResults.txt','',1,0); % skip 1 rows and 0 columns

%% Reference Values for dimensionless values
% Reference velocity: Fibre B starting velocity
vB1=[Numeric(1,14) Numeric(1,15) Numeric(1,16)];
vRef=max(norm(vB1),norm(vB1));
if vRef==0;
    vRef=1;
end

% Reference angular velocity: Fibre B starting angular velocity
omB1=[Numeric(1,20) Numeric(1,21) Numeric(1,22)];
omeRef=max(norm(omB1),norm(omB1));
if omeRef==0;
    omeRef=1;
end

% Reference Energy: tot Energy at Start
ERef= Numeric(1,28);

% Reference linear Momentum: tot linear Momentum at Start
MomRef= Numeric(1,32);

%% Compare Results
cd (curDir)

% Fibre A = Fibre 1 in LIGGGHTS
Name='Fibre1_Velo';
angles=Analytic(:,1);
vA2_Analytic=[Analytic(:,2:4)]./vRef;
vA2_Numeric =[Numeric(:,5:7)]./vRef;
```

```
    CollisionAll.ERot1(i,:)=ERotA1(i,:)+ERotB1(i,:);

    ERotA2(i,:)=1/2*IA.*dot(CollisionAll.omA2(i,:),CollisionAll.omA2(i,:));
    ERotB2(i,:)=1/2*IB.*dot(CollisionAll.omB2(i,:),CollisionAll.omB2(i,:));
    CollisionAll.ERot2(i,:)=ERotA2(i,:)+ERotB2(i,:);

    % Total kinetic Energy
    CollisionAll.EKin1(i,:)=CollisionAll.ETrans1(i,:)+CollisionAll.ERot1(i,:);
    CollisionAll.EKin2(i,:)=CollisionAll.ETrans2(i,:)+CollisionAll.ERot2(i,:);

    %% Momentum in system
    % Linear Momentum P=m*v=const.
    PA1(i,:)=mA*vA1;
    PB1(i,:)=mB*vB1;
    CollisionAll.P1(i,:)=norm(PA1(i,:)+PB1(i,:));

    PA2(i,:)=mA*vA2(i,:);
    PB2(i,:)=mB*vB2(i,:);
    CollisionAll.P2(i,:)=norm(PA2(i,:)+PB2(i,:));

end


% 4) Save Data in .txt -File
% Write analytical results in .txt -file
curDir=pwd;
cd ..
cd ..
MainDir=pwd;
cd (curDir)

fileName='AnalyticResults.txt';
AnalyticRes = fopen(fullfile(MainDir,'postProcessing',fileName),'w');

fprintf(AnalyticRes,'%s',...
   '(1)alpha (2)vA2x (3)vA2y (4)vA2z (5)VB2x (6)VB2y (7)VB2z (8)omA2x (9)omA2y (10)omA2z (11)omB2x (12)omB2y (13)omB2z (14)EKin1 (15)EKin2 (16)P1 (17)P2 (18)CoR' );

for i=1:length(alpha)
fprintf(AnalyticRes,...
'\n %4.2f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f',...
 alpha(i)/pi*180,...
 CollisionAll.vA2(i,1), CollisionAll.vA2(i,2), CollisionAll.vA2(i,3),...
 CollisionAll.vB2(i,1), CollisionAll.vB2(i,2), CollisionAll.vB2(i,3),...
 CollisionAll.omA2(i,1), CollisionAll.omA2(i,2), CollisionAll.omA2(i,3),...
 CollisionAll.omB2(i,1), CollisionAll.omB2(i,2), CollisionAll.omB2(i,3),...
 CollisionAll.EKin1(i),CollisionAll.EKin2(i),...
 CollisionAll.P1(i),CollisionAll.P2(i),CollisionAll.CoR);
end
fclose(AnalyticRes);

end
```

```
plotVeloStatus=func_plotVelo(Name,angles,vA2_Analytic,vA2_Numeric,CoR, interpreterName);

Name='Fibre1_angularVelo';
angles=Analytic(:,1);
omA2_Analytic=[Analytic(:,9)]./omeRef;
omA2_Numeric =[Numeric(:,12)]./omeRef;
plotVeloStatus=func_plotAngVelo(Name,angles,omA2_Analytic,omA2_Numeric,CoR, interpreterName);

% Fibre B = Fibre 2 in LIGGGHTS
Name='Fibre2_Velo';
angles=Analytic(:,1);
vB2_Analytic=[Analytic(:,5:7)]./vRef;
vB2_Numeric =[Numeric(:,17:19)]./vRef;
plotVeloStatus=func_plotVelo(Name,angles,vB2_Analytic,vB2_Numeric,CoR, interpreterName);

Name='Fibre2_angularVelo';
angles=Analytic(:,1);
omB2_Analytic=[Analytic(:,12)]./omeRef;
omB2_Numeric =[Numeric(:,24)]./omeRef;
plotVeloStatus=func_plotAngVelo(Name,angles,omB2_Analytic,omB2_Numeric,CoR, interpreterName);

% Compare Energy Conservation
   Name='EnergyConservation';

   % Analytical Results:
   Etot_Anal1 = Analytic(:,14)./ERef; % EKin1
   Etot_Anal2 = Analytic(:,15)./ERef; % EKin2

   % Numerical Results:
   Etot_num1 = Numeric(:,28)./ERef;
   Etot_num2 = Numeric(:,31)./ERef;

   plotVeloStatus=func_plotEng(Name,angles,Etot_Anal1,Etot_Anal2,...
       Etot_num1,Etot_num2,CoR,interpreterName);


% Compare Momentum Conservation
   Name='LinearMomentumConservation';

   % Analytical Results:
   Momtot_Anal1 = Analytic(:,16)./MomRef; % Mom1
   Momtot_Anal2 = Analytic(:,17)./MomRef; % Mom2

   % Numerical Results:
   Momtot_num1 = Numeric(:,32)./MomRef;
   Momtot_num2 = Numeric(:,34)./MomRef;

   plotVeloStatus=func_plotMom(Name,angles,Momtot_Anal1,Momtot_Anal2,...
       Momtot_num1,Momtot_num2,CoR,interpreterName);

disp('End of program! Have fun ...!')

```

```
% Format
rect=[10 10 600 600];

% If MATLAb is used for plots
fontSizeTitle=16;
fontSizeAxis=18;

lineWidth=2;
markerSize    = 9;
stdTextFontSize  = 18;
labelFontSize    = 22;

preAfterSymbol = '';
if(  strcmp(interpreterName, 'latex') )
    preAfterSymbol = '$';
end

```

```matlab
function plotVeloStatus=func_plotVelo(Name,angles,omAnalytic,omNumeric,...
    CoR, interpreterName)
global showFig resultDir
% Plot velocity data of fibre determined by an analytical and by an
% numerical solution

%% Formating
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

% Get Screen Size
scrsz = get(0,'ScreenSize');

% Create plot
fig=figure('Name',Name,'Position',rect,...
    'Visible',showFig); % [left, bottom, width, height]
 plot(angles(:,1),omAnalytic(:,1),'-ro',...
     'LineWidth',lineWidth,'markersize',markerSize*0.8); hold on; % omy

 plot(angles(:,1),omNumeric(:,1),'-.bx',...
     'LineWidth',lineWidth,'markersize',markerSize); hold on; % omy

 legendStringAngVelo={[preAfterSymbol,'\omega_{y,anal.}^{+}',preAfterSymbol],...
        [preAfterSymbol,'\omega_{y,num.}^{+}',preAfterSymbol]};

 legend(legendStringAngVelo,...
       'Location','southeast','Orientation','horizontal')

 if(  strcmp(interpreterName, 'latex') )
  legend(legendStringAngVelo,...
       'Location','southeast','Orientation','vertical',...
       'FontWeight','bold',...
       'interpreter',interpreterName,'FontSize',labelFontSize)
 end

xlabel([preAfterSymbol,'\alpha (degree)',preAfterSymbol],'interpreter',interpreterName)
ylabel([preAfterSymbol,'\omega_{y}^{+}',preAfterSymbol],'interpreter',interpreterName)

title({['Compare analytical and numerical results CoR = ',num2str(CoR)]},...
    'interpreter',interpreterName,'FontSize',fontSizeTitle)

grid off;
box on;
legend boxoff;
xlim([-90 0]);
set(gca,'XTick',[-90:15:0]);
set(gca,'XMinorTick','on','YMinorTick','on');
set(0,'defaultaxesfontsize',labelFontSize);
set(0,'defaulttextfontsize',labelFontSize);
set(gca,'FontSize',labelFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',labelFontSize); set(ylhand,'fontsize',labelFontSize)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

% Save Figures
% saveas(gcf,strcat(FigDir, Name), 'fig');
% set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 18])
% print('-dpdf','-r300', [Name])
curDir=pwd;
cd ..
cd ..
print('-dpng','-r300', [resultDir, Name])
cd (curDir)
plotVeloStatus=1;
end

```

```
function plotVeloStatus=func_plotEng(Name,angles,Etot_Anal1,Etot_Anal2,...
    Etot_num1,Etot_num2,CoR, interpreterName)
global showFig resultDir
% Plot energy data of fibre determined by an analytical and by an
% numerical solution

%% Formating
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

%% Formating
fontSizeTitle=16;
fontSizeAxis=20;
fontSizeLabel=24;
markerSize = 9;
lineWidth=2;

% Get Screen Size
scrsz = get(0,'ScreenSize');

% Create plot
fig=figure('Name',Name,'Position',rect,...
    'Visible',showFig); % [left, bottom, width, height]

 plot(angles(:,1),Etot_Anal1(:),'-ro',...
     'LineWidth',lineWidth,'MarkerSize',markerSize*0.8); hold on; % Etot1_anal

 plot(angles(:,1),Etot_Anal2(:),'-rx',...
     'LineWidth',lineWidth,'MarkerSize',markerSize*0.8); hold on; % Etot2_anal

 plot(angles(:,1),Etot_num1(:),'-.bo',...
     'LineWidth',lineWidth,'MarkerSize',markerSize); hold on; % Etot1_num

 plot(angles(:,1),Etot_num2(:),'-.bx',...
     'LineWidth',lineWidth,'MarkerSize',markerSize); hold on; % Etot2_num

  legendStringEng={[preAfterSymbol,'E_{tot,1,anal.}^{+}',preAfterSymbol], ...
        [preAfterSymbol,'E_{tot,2,anal.}^{+}',preAfterSymbol],...
        [preAfterSymbol,'E_{tot,1,num.}^{+}',preAfterSymbol], ...
        [preAfterSymbol,'E_{tot,2,num.}^{+}',preAfterSymbol]};

  legend(legendStringEng,...
      'Location','southwest','Orientation','vertical')

 if(  strcmp(interpreterName, 'latex') )
  legend(legendStringEng,...
      'Location','best','Orientation','vertical',...
      'FontWeight','bold',...
      'interpreter',interpreterName,'FontSize',fontSizeLabel)
 end

xlabel([preAfterSymbol,'\alpha (degree)',preAfterSymbol],'interpreter',interpreterName)
ylabel([preAfterSymbol,'E_{tot,1}^{+},  E_{tot,2}^{+}',preAfterSymbol],'interpreter',interpreterName)

title({['Compare analytical and numerical results CoR = ',num2str(CoR)]},...
    'interpreter',interpreterName,'FontSize',fontSizeTitle)

grid off;
box on;
legend boxoff;
xlim([-90 0]);
set(gca,'XTick',[-90:15:0]);
set(gca,'XMinorTick','on','YMinorTick','on');
set(0,'defaultaxesfontsize',fontSizeAxis);
set(0,'defaulttextfontsize',fontSizeAxis);
set(gca,'FontSize',fontSizeAxis);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',fontSizeLabel); set(ylhand,'fontsize',fontSizeLabel)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

% Save Figures
% saveas(gcf,strcat(FigDir, Name), 'fig');
% set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 18])
% print('-dpdf','-r300', [Name])
curDir=pwd;
cd ..
cd ..
print('-dpng','-r300', [resultDir,Name])
cd (curDir)
plotVeloStatus=1;
end
```

```
function plotVeloStatus=func_plotE og (Nag e'm an, les'm E og tot_Anal1'm E og tot_Anal2'm...
    E og tot_nug 1'm E og tot_nug 2'm CoR'm interpreterNag e)
, lobal showFi, resultDir
% Plot ener, y data of fibre deterg ined by an analytical and by an
% nug erical solution

%% Forg atin,
run('forg attin, _E AMTAL.g ')B
if( strcg p(interpreterNag e'm 'te9') )
  run('forg attin, _0NG.g ')B
end

% 0et Screen SiUe
scrsU= , et(x'm 'ScreenSiUe')B

% Create plot
fi, =fi, ure('Nag e''m Nag e'm 'Position''m rect'm...
   'Visible''m showFi, )B% z left'm bottog 'm width'm hei, htk

 plot(an, les(W1)'m E og tot_Anal1(W'm '[ro''m...
    'Tine]  idth''m line]  idth'm 'g ar: ersiUe''m g ar: erSiUe-x.8)B hold onB% *tot1_anal

 plot(an, les(W1)'m E og tot_Anal2(W'm '[r9''m...
    'Tine]  idth''m line]  idth'm 'g ar: ersiUe''m g ar: erSiUe-x.8)B hold onB% *tot2_anal

 plot(an, les(W1)'m E og tot_nug 1(W'm '[.bo''m...
    'Tine]  idth''m line]  idth'm 'g ar: ersiUe''m g ar: erSiUe)B hold onB% *tot1_nug

 plot(an, les(W1)'m E og tot_nug 2(W'm '[.b9''m...
    'Tine]  idth''m line]  idth'm 'g ar: ersiUe''m g ar: erSiUe)B hold onB% *tot2_nug

 le, enStrin, E og ={z preAfterSyg bol'm 'P_{tot'm 1'm anal.}^{+}''m preAfterSyg bolk'm...
        z preAfterSyg bol'm 'P_{tot'm 2'm anal.}^{+}''m preAfterSyg bolk'm...
        z preAfterSyg bol'm 'P_{tot'm 1'm nug .}^{+}''m preAfterSyg bolk'm...
        z preAfterSyg bol'm 'P_{tot'm 2'm nug .}^{+}''m preAfterSyg bolk}B

 le, end(le, enStrin, E og 'm...
       'Tocation''m southwest''m Orientation''m vertical')


 if( strcg p(interpreterNag e'm 'late9') )
le, end(le, enStrin, E og 'm...
       'Tocation''m best''m Orientation''m vertical''m...
       'Font]  ei, ht''m bold''m...
       'interpreter''m interpreterNag e'm 'FontSiUe''m labelFontSiUe)
 end


9label(z preAfterSyg bol'm '\alpha (de, ree)''m preAfterSyg bolk'm 'interpreter''m interpreterNag e)
ylabel(z preAfterSyg bol'm 'P_{tot'm 1}^{+}'m P_{tot'm 2}^{+}''m preAfterSyg bolk'm 'interpreter''m interpreterNag e)

title({z 'Cog pare analytical and nug erical results CoR = ''m nug 2str(CoR)k}'m...
    'interpreter''m interpreterNag e'm 'FontSiUe''m fontSiUeMtitle)

9g in=g in(an, les)B
9g a9=g a9(an, les)B

yg in=g in(g in(zE og tot_Anal1'm E og tot_Anal2'm E og tot_nug 1'm E og tot_nug 2k))B
yg a9=g a9(g a9(zE og tot_Anal1'm E og tot_Anal2'm E og tot_nug 1'm E og tot_nug 2k))B

a9is(z9g in 9g a9 x 1.x1k)

, rid offB
bo9 onB
le, end bo9offB
set(, ca'm 'XMlc: ''m z[; x W15Wk)B
set(, ca'm 'XE inorMlc: ''m on''m 'YE inorMlc: ''m on')B
set(x'm 'defaulta9esfontsiUe''m stdMe9tFontSiUe)B
set(x'm 'defaultte9tfontsiUe''m stdMe9tFontSiUe)B
set(, ca'm 'FontSiUe''m stdMe9tFontSiUe)B
set(, ca'm 'Font]  ei, ht''m norg al')B
9lhand = , et(, ca'm '9label')B ylhand = , et(, ca'm 'ylabel')B
set(9lhand'm 'fontsiUe''m labelFontSiUe)B set(ylhand'm 'fontsiUe''m labelFontSiUe)
set(9lhand'm 'FontNag e''m 'Mlg es')B set(9lhand'm 'FontAn, le''m italic')B
set(9lhand'm 'Font]  ei, ht''m bold')B
set(ylhand'm 'FontNag e''m 'Mlg es')B set(ylhand'm 'FontAn, le''m italic')B
set(ylhand'm 'Font]  ei, ht''m bold')B
set(, cf'm 'paperunits''m centig eters')

% Save Fi, ures
% saveas(, cf'm strcat(Fi, Dir'm Nag e)'m 'fi, ')B
% set(, cf'm 'paperunits''m centig eters''m paperposition''m zx x 21 18k)
% print('[dpdf''m '[r3xx''m zNag ek)
curDir=pwdB
cd ..
cd ..
print('[dpn, ''m '[r3xx''m zresultDir'm Nag ek)
cd (curDir)

plotVeloStatus=1B
end
```

```
function plotVeloStatus=func_plotVelo(Name,angles,Ay nalr tic,ANume. ic,22
    Ro) , inte. p. ete. Name3
glohal swoF Dg .esult4 i.
P  vlot Aelocitr data of fih. e dete. mined hr an analr tical and hr an
P  nume. ical solution

P P  Do. mating
. un('fo. matting_My TLy B2m'3
if(  st. cmp(inte. p. ete. Name, 'tex'33
   . un('fo. matting_GNU2m'3
end

P  Get Sc. een Size
sc. sz = get(0,'Sc. eenSize'3

P  R. eate plot
fig=figu. e('Name',Name,'vosition',. ect,22
    'Visihle',swoF Dg3 P  [left, hottom, F idtw, weigwt]
 plot(angles(:,13,Ay nalr tic(:,13'-. o',22
     'LineWidtw',lineWidtw,'ma. ke. Size',ma. ke. Size*0283 wold on; P  Ax
 plot(angles(:,13,Ay nalr tic(:,b3'-. x',22
     'LineWidtw',lineWidtw,'ma. ke. Size',ma. ke. Size*0283 wold on; P  Az
 plot(angles(:,13ANume. ic(:,13'-2ho',22
     'LineWidtw',lineWidtw,'ma. ke. Size',ma. ke. Size3 wold on; P  Ax
 plot(angles(:,13ANume. ic(:,b3'-2hx',22
     'LineWidtw',lineWidtw,'ma. ke. Size',ma. ke. Size3 wold on; P  Az

legendSt. ing={[p. ey fte. Sr mhol,'A_{x,anal2} {^\',p. ey fte. Sr mhol], 22
        [p. ey fte. Sr mhol,'A_{z,anal2} {^\',p. ey fte. Sr mhol],22
        [p. ey fte. Sr mhol,'A_{x,num2} {^\', p. ey fte. Sr mhol], 22
        [p. ey fte. Sr mhol,'A_{z,num2} {^\', p. ey fte. Sr mhol]\;

legend(legendSt. ing,'Location','F est','+ . ientation','Ae. tical'3


 if(  st. cmp(inte. p. ete. Name, 'latex'33
   legend(legendSt. ing,22
       'Location','F est','+ . ientation','Ae. tical',22
       'DontWeigwt','hold',22
       'inte. p. ete. ',inte. p. ete. Name,'DontSize',lahelDontSize3
 end

xlahel([p. ey fte. Sr mhol,'Galpwa (deg. ee3,p. ey fte. Sr mhol],'inte. p. ete. ',inte. p. ete. Name3
r lahel([p. ey fte. Sr mhol,'A_{x\} {^\, A_{z\} {^\',p. ey fte. Sr mhol],'inte. p. ete. ',inte. p. ete. Name3

title({['Rompa. e analr tical and nume. ical . esults Ro)  = ',numCst. (Ro) 3]\,22
    'inte. p. ete. ',inte. p. ete. Name,'DontSize',fontSizeTitle3

g. id off;
hox on;
legend hoxoff;
xlim([-90 0]3
set(gca,'XTick',[-90:15:0]3
set(gca,'XMino. Tick','on','YMino. Tick','on'3
set(0,'defaultaxesfontsize',lahelDontSize3
set(0,'defaulttextfontsize',lahelDontSize3
set(gca,'DontSize',lahelDontSize3
set(gca,'DontWeigwt','no. mal'3
xlwand = get(gca,'xlahel'3r lwand = get(gca,'r lahel'3
set(xlwand,'fontsize',lahelDontSize3 set(r lwand,'fontsize',lahelDontSize3
set(xlwand,'DontName','Times'3 set(xlwand,'Donty ngle','italic'3
set(xlwand,'DontWeigwt','hold'3
set(r lwand,'DontName','Times'3 set(r lwand,'Donty ngle','italic'3
set(r lwand,'DontWeigwt','hold'3
set(gcf, 'pape. units', 'centimete. s'3

P  SaAe Digu. es
P  saAeas(gcf,st. cat(Dig4 i., Name3 'fig'3
P  set(gcf, 'pape. units', 'centimete. s', 'pape. position', [0 0 C1 18]3
P  p. int('-dpdf','-. b00', [Name]3
cu. 4 i. =pF d;
cd 22
cd 22
p. int('-dpng','-. b00', [. esult4 i.,Name]3
cd (cu. 4 i. 3
plotVeloStatus=1;
end

```

```
%% postProcess Numeric (LIGGGHTS(R)) results
% Read Data and save Data of all Cases in single txt File
clear all; close all; clc

curDir=pwd;
cd ..
cd ..
MainDir=pwd;
cd postProcessing

fileEng=dir('energy_Case*');
nCases=length(fileEng);

for i=1:1:nCases
    Data.Num(i,1)=i;

    % Velocity and Angular velocty
    % Fibre A
    fileEng1=dir('velocityOrientation1_Case_*');
    tempD= dlmread(fileEng1(i).name,'',1,0);
    Data.Num(i,2)=tempD(1,2); % vA1x @ Start
    Data.Num(i,3)=tempD(1,3); % vA1y @ Start
    Data.Num(i,4)=tempD(1,4); % vA1z @ Start

    Data.Num(i,5)=tempD(end,2); % vA2x @ End
    Data.Num(i,6)=tempD(end,3); % vA2y @ End
    Data.Num(i,7)=tempD(end,4); % vA2z @ End

    Data.Num(i,8)=tempD(1,5); % omA1x @ Start
    Data.Num(i,9)=tempD(1,6); % omA1y @ Start
    Data.Num(i,10)=tempD(1,7); % vA1z @ Start

    Data.Num(i,11)=tempD(end,5); % omA2x @ End
    Data.Num(i,12)=tempD(end,6); % omA2y @ End
    Data.Num(i,13)=tempD(end,7); % omA2z @ End

    % Fibre B
    fileEng2=dir('velocityOrientation2_Case_*');
    tempD= dlmread(fileEng2(i).name,'',1,0);
    Data.Num(i,14)=tempD(1,2); % vB1x @ Start
    Data.Num(i,15)=tempD(1,3); % vB1y @ Start
    Data.Num(i,16)=tempD(1,4); % vB1z @ Start

    Data.Num(i,17)=tempD(end,2); % vB2x @ End
    Data.Num(i,18)=tempD(end,3); % vB2y @ End
    Data.Num(i,19)=tempD(end,4); % vB2z @ End

    Data.Num(i,20)=tempD(1,5); % omB1x @ Start
    Data.Num(i,21)=tempD(1,6); % omB1y @ Start
    Data.Num(i,22)=tempD(1,7); % vB1z @ Start

    Data.Num(i,23)=tempD(end,5); % omB2x @ End
    Data.Num(i,24)=tempD(end,6); % omB2y @ End
    Data.Num(i,25)=tempD(end,7); % omB2z @ End

    % Energy Data
    fileEng3=dir('energy_Case_*');
    tempD= dlmread(fileEng3(i).name,'',1,0);
    Data.Num(i,26)=tempD(1,2); % Ekin @ Start
    Data.Num(i,27)=tempD(1,3); % Epot @ Start
    Data.Num(i,28)=tempD(1,4); % Etot @ Start

    Data.Num(i,29)=tempD(end,2); % Ekin @ End
    Data.Num(i,30)=tempD(end,3); % Epot @ End
    Data.Num(i,31)=tempD(end,4); % Etot @ End

    % Momentum Data
    fileEng4=dir('momentum_Case_*');
    tempD= dlmread(fileEng4(i).name,'',1,0);
    Data.Num(i,32)=tempD(1,4); % LinMomTot @ Start
    Data.Num(i,33)=tempD(1,9); % AngMomTot @ Start

    Data.Num(i,34)=tempD(end,4); % LinMomTot @ End
    Data.Num(i,35)=tempD(end,9); % AngMomTot @ End

end
cd (curDir)

%% Save Data in .txt -File
% Write numerical results in .txt -file

fileName='NumericResults.txt';
NumRes = fopen(fullfile(MainDir,'postProcessing',fileName),'w');

fprintf(NumRes,'%s',...
  'nCase vAx1 vAy1 vAz1 vAx2 vAy2 vAz2 omAx1 omAy1 omAz1 omAx2 omAy2 omAz2 vBx1 vBy1 vBz1 vBx2 vBy2 vBz2 omBx1 omBy1↙
omBz1 omBx2 omBy2 omBz2 Ekin1 Epot1 Etot1 Ekin2 Epot2 Etot2 LinMomTot1 AngMomTot1 LinMomTot2 AngMomTot2');
for i=1:nCases ;
fprintf(NumRes,...
'\n %1i %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f↙
%8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f ',...
 Data.Num(i,:));
end
fclose(NumRes);

disp('Numerical results have been saved in:');
disp('postProcessing/NumericResults.txt');
```

```
% Programm to check Energy conservation
clc; close all; clear all;

%% General settings
curDir=pwd;
showFig='off';          % show plots as figures
SingleResultDir='./postProcessing/SingleResults/';
interpreterName='tex';    % tex(GNU) or latex(MATLAB)

%% End of user input
cd ..
cd ..
cd ./postProcessing

if exist('SingleResults') == 0 % if folder does not exist create it!
   mkdir('SingleResults')
end

filename='energy.dat';
cd ..
pathname=['./post/'];
Data.All = dlmread(strcat(pathname,filename),'',1,0); % skip 1 rows and 0 columns
cd (curDir)

% Colum division
Data.timeStep=Data.All(:,1);
Data.kinEnergy=Data.All(:,2);
Data.potEnergy=Data.All(:,3);
Data.totEnergy=Data.All(:,4);

% Formating
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

%% Plot Energy Conservation
fig1=figure('Name','Energy Conservation','Visible',showFig);
plot(Data.timeStep,Data.totEnergy,'r','LineWidth',lineWidth*2);hold on;
plot(Data.timeStep,Data.kinEnergy,'b','LineWidth',lineWidth*0.8);hold on;
plot(Data.timeStep,Data.potEnergy,'k','LineWidth',lineWidth);hold on;

xlabel([preAfterSymbol,'time',preAfterSymbol],'interpreter',interpreterName);
ylabel([preAfterSymbol,'Energy',preAfterSymbol],'interpreter',interpreterName)

legendString={[preAfterSymbol,'E_{tot}',preAfterSymbol],...
          [preAfterSymbol,'E_{kin}',preAfterSymbol],...
          [preAfterSymbol,'E_{pot}',preAfterSymbol]};

legend(legendString,'Location','west','Orientation','vertical')

 if (strcmp(interpreterName, 'latex'))
   legend(legendString,...
       'Location','best','Orientation','vertical',...
       'FontWeight','bold',...
       'interpreter',interpreterName,'FontSize',labelFontSize)
 end

grid off
box on
legend boxoff
set(gca,'XMinorTick','on','YMinorTick','on');
set(0,'defaultaxesfontsize',labelFontSize);
set(0,'defaulttextfontsize',labelFontSize);
set(gca,'FontSize',labelFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',labelFontSize); set(ylhand,'fontsize',labelFontSize)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

% Print Plot to pdf
cd ..
cd ..
set(gcf,'paperunits','centimeter','paperposition',[0 0 21 18])
print('-dpng','-r300', [SingleResultDir,'EngConservation'])
cd (curDir)

%% Plot tot. Energy Deviation from Start
EtotRef=Data.totEnergy(1);
for i=1:1:length(Data.totEnergy);
Data.ETotDeviation(i)=(Data.totEnergy(i)-Data.totEnergy(1))/...
    Data.totEnergy(1);
end
fig2=figure('Name','Energy deviation from initial Etot','Visible',showFig);
plot(Data.timeStep,Data.ETotDeviation,'k',...
     'LineWidth',lineWidth);
xlabel([preAfterSymbol,'time',preAfterSymbol],'interpreter',interpreterName);
ylabel([preAfterSymbol,'Deviation',preAfterSymbol],'interpreter',interpreterName)

title({'Deviation from initial Etot [-]'},...
    'interpreter',interpreterName,'FontSize',labelFontSize*0.8)


grid off
box on
legend boxoff
set(gca,'XMinorTick','on','YMinorTick','on');
set(0,'defaultaxesfontsize',labelFontSize);
set(0,'defaulttextfontsize',labelFontSize);
set(gca,'FontSize',labelFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',labelFontSize); set(ylhand,'fontsize',labelFontSize)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

% Print Plot to pdf
cd ..
cd ..
set(gcf,'paperunits','centimeter','paperposition',[0 0 21 18])
print('-dpng','-r300', [SingleResultDir,'EnergyDerivation'])
cd (curDir)

disp('Energy conservation done.')

```

```
% Programm to check Energy conservation
clc; close all; clear all;

%% General settings
curDir=pwd;
showFig='off';          % show plots as figures
SingleResultDir='./postProcessing/SingleResults/';
interpreterName='tex';    % tex(GNU) or latex(MATLAB)

%% End of user input
cd ..
cd ..
cd ./postProcessing

if exist('SingleResults') == 0 % if folder does not exist create it!
   mkdir('SingleResults')
end

filename='momentum.dat';
cd ..
pathname=['./post/'];
Data.All = dlmread(strcat(pathname,filename),'',1,0); % skip 1 rows and 0 columns
cd (curDir)

% Colum division
Data.timeStep=Data.All(:,1);
Data.LinMom1=Data.All(:,2);
Data.LinMom2=Data.All(:,3);
Data.LinMomTot=Data.All(:,4);
Data.AngMomSpin1=Data.All(:,5);
Data.AngMomSpin2=Data.All(:,6);
Data.AngMomRot1=Data.All(:,7);
Data.AngMomRot2=Data.All(:,8);
Data.AngMomTot=Data.All(:,9);

% Formating
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

%% Linear Momentum
% Plot Linear Momentum Conservation
fig1=figure('Name','Linear Momentum Conservation','Visible',showFig);
plot(Data.timeStep,Data.LinMom1,'b','LineWidth',lineWidth*1.2);hold on;
plot(Data.timeStep,Data.LinMom2,'k','LineWidth',lineWidth*1.0);hold on;
plot(Data.timeStep,Data.LinMomTot,'r','LineWidth',lineWidth*0.8);hold on;

xlabel([preAfterSymbol,'time',preAfterSymbol],'interpreter',interpreterName);
ylabel([preAfterSymbol,'Linear Momentum',preAfterSymbol],'interpreter',interpreterName)

legendString={[preAfterSymbol,'LinMom_{1}',preAfterSymbol],...
         [preAfterSymbol,'LinMom_{2}',preAfterSymbol],...
         [preAfterSymbol,'LinMom_{tot}',preAfterSymbol]};

legend(legendString,'Location','west','Orientation','vertical')

 if (strcmp(interpreterName, 'latex'))
   legend(legendString,...
       'Location','best','Orientation','vertical',...
       'FontWeight','bold',...
       'interpreter',interpreterName,'FontSize',labelFontSize)
 end

grid off
box on
legend boxoff
set(gca,'XMinorTick','on','YMinorTick','on');
set(0,'defaultaxesfontsize',labelFontSize);
set(0,'defaulttextfontsize',labelFontSize);
set(gca,'FontSize',labelFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',labelFontSize); set(ylhand,'fontsize',labelFontSize)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

% Print Plot to pdf
cd ..
cd ..
set(gcf,'paperunits','centimeter','paperposition',[0 0 21 18])
print('-dpng','-r300', [SingleResultDir,'LinMom'])
cd (curDir)

% Plot Linear Momentum Deviation from Start
for i=1:1:length(Data.LinMomTot);
Data.LinMomDeviation(i)=(Data.LinMomTot(i)-Data.LinMomTot(1))/...
    Data.LinMomTot(1)*100;
end
fig2=figure('Name',...
'Linear Momentum deviation from initial Linear Momentum','Visible',showFig);
plot(Data.timeStep,Data.LinMomDeviation,'k',...
    'LineWidth',lineWidth);
xlabel([preAfterSymbol,'time',preAfterSymbol],'interpreter',interpreterName);
ylabel([preAfterSymbol,'Deviation',preAfterSymbol],...
    'interpreter',interpreterName)

title({'Deviation from initial LinMom [-]'},...
    'interpreter',interpreterName,'FontSize',labelFontSize*0.8)

grid off
box on
legend boxoff
set(gca,'XMinorTick','on','YMinorTick','on');
set(0,'defaultaxesfontsize',labelFontSize);
set(0,'defaulttextfontsize',labelFontSize);
set(gca,'FontSize',labelFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',labelFontSize); set(ylhand,'fontsize',labelFontSize)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

% Print Plot to pdf
cd ..
cd ..
set(gcf,'paperunits','centimeter','paperposition',[0 0 21 18])
print('-dpng','-r300', [SingleResultDir,'LinMomDerivation'])
cd (curDir)

%% Angular Momentum
% Plot Angular Momentum Conservation
fig3=figure('Name','Angular Momentum Conservation','Visible',showFig);
plot(Data.timeStep,Data.AngMomTot,'r','LineWidth',lineWidth*2);hold on;
plot(Data.timeStep,Data.AngMomSpin1,'b*','LineWidth',lineWidth*1.1);hold on;
plot(Data.timeStep,Data.AngMomSpin2,'b','LineWidth',lineWidth*0.6);hold on;
plot(Data.timeStep,Data.AngMomRot1,'k*','LineWidth',lineWidth*0.6);hold on;
plot(Data.timeStep,Data.AngMomRot2,'k','LineWidth',lineWidth*1);hold on;

xlabel([preAfterSymbol,'time',preAfterSymbol],'interpreter',interpreterName);
ylabel([preAfterSymbol,'Angular Momentum',preAfterSymbol],'interpreter',interpreterName);

legendString={[preAfterSymbol,'AngMomRot_{tot}',preAfterSymbol],...
         [preAfterSymbol,'AngMomSpin_{1}',preAfterSymbol],...
```

```
        [preAfterSymbol,'AngMomSpin_{2}',preAfterSymbol],...
        [preAfterSymbol,'AngMomRot_{1}',preAfterSymbol],...
        [preAfterSymbol,'AngMomRot_{2}',preAfterSymbol]};

legend(legendString,'Location','west','Orientation','vertical')

 if (strcmp(interpreterName, 'latex'))
   legend(legendString,...
      'Location','best','Orientation','vertical',...
      'FontWeight','bold',...
      'interpreter',interpreterName,'FontSize',labelFontSize)
 end

grid off
box on
legend boxoff
set(gca,'XMinorTick','on','YMinorTick','on');
set(0,'defaultaxesfontsize',labelFontSize);
set(0,'defaulttextfontsize',labelFontSize);
set(gca,'FontSize',labelFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',labelFontSize); set(ylhand,'fontsize',labelFontSize)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

 % Print Plot to pdf
cd ..
cd ..
set(gcf,'paperunits','centimeter','paperposition',[0 0 21 18])
print('-dpng','-r300', [SingleResultDir,'AngMom'])
cd (curDir)

% Plot Angular Momentum Deviation from Start
for i=1:1:length(Data.AngMomTot);
Data.AngMomDeviation(i)=(Data.AngMomTot(i)-Data.AngMomTot(1))/Data.AngMomTot(1)*100;
end
fig4=figure('Name',...
'Linear Momentum deviation from initial Linear Momentum','Visible',showFig);
plot(Data.timeStep,Data.AngMomDeviation,'k','LineWidth',lineWidth);

xlabel([preAfterSymbol,'time',preAfterSymbol],'interpreter',interpreterName);
ylabel([preAfterSymbol,'Deviation',preAfterSymbol],...
    'interpreter',interpreterName)

title({'Deviation from initial AngMom [-]'},...
    'interpreter',interpreterName,'FontSize',labelFontSize*0.8)

grid off
box on
legend boxoff
set(gca,'XMinorTick','on','YMinorTick','on');
set(0,'defaultaxesfontsize',labelFontSize);
set(0,'defaulttextfontsize',labelFontSize);
set(gca,'FontSize',labelFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',labelFontSize); set(ylhand,'fontsize',labelFontSize)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

disp('Momentum conservation done.')

% Print Plot to pdf
cd ..
cd ..
set(gcf,'paperunits','centimeter','paperposition',[0 0 21 18])
print('-dpng','-r300', [SingleResultDir,'AngMomDerivation'])
cd (curDir)


```

```
Input Data: Deform Box - Fibre Flock Generation - TRI-AXIAL phi=0.8%
======================================================================
compactEveryNthDEMStep ... dtContact/dtDEM
dampingNormal          ... small damping; 5e-2, normal direction to gently remove kinetic energy; coefficient of restitution = 1 sets the↙
damping to 0
dampingTang            ... set to zero
DefMod                 ... deformation mode 1 ... uni-axial 3 ... tri-axial
dMajor                 ... length of fibre
dMinor                 ... width of fibre, for roughness all fibres need same dMinor!
dQr,i                  ... 2D fibre distribution
dtDEM                  ... Time Step size of DEM simulation, adjust acc. to Luding: Introduction to DEM dtDEM=tc/50
finalBox               ... final  (post-deformation) box dimensions X Y Z
frictionCoeff          ... set to zero
initBox                ... initial(pre-deformation) box dimensions X Y Z
k_overlap              ... max. allowed overlap at single deformation/compression
liquidDynViscosity     ... lubrication model for fibre/fibre interaction if zero -> no lubr.
phiFib                 ... fibre volume fraction
r                      ... distribution 0 or 3
rhoFib                 ... density of all fibres
rhoFluid               ... density of fluid
roughFact              ... fraction of dMinor giving the particle roughness (will model fibrils on fibre), can not be zero!
roughStiffFact         ... fraction of the fibre normal stiffness used to calculate the repelling force due to roughness contact
stiffnessNormal        ... is taken here  for the spring constant for the normal contact
stiffnessTang          ... set to zero
dMinorRealFact         ... Factor for dMajor value that is written in output. output_dMinor=dMinor/dMinorRealFact

DEM PARAMETER:
==============
roughFact     roughStiffFact  frictionCoeff   liquidDynViscosity
  1e-2           1e-3             0.0              0.0

dtDEM   k_overlap   compactEveryNthDEMStep
1e-6      0.005                 50

BOX DIMENSIONS:
===============
initBoxX initBoxY initBoxY
2.0      2.0      2.0

finalBoxX finalBoxY finalBoxY
1.0       1.0       1.0

FIBRE/FLUID DATA & DEFORMATION MODE:
=====================================
r  rhoFib  rhoFluid  phiFib DefMod
0  1.27    1.0       0.008   3

dMajor      dMinor     dQr,i    stiffnessNormal dampingNormal  stiffnessTang   dampingTang dMinorRealFact
0.040       0.020      0.20       2e4          1e-2           0.0          0.0           1.0
0.100       0.020      0.20       2e4          1e-2           0.0          0.0           1.0
0.200       0.020      0.20       2e4          1e-2           0.0          0.0           1.0
0.300       0.020      0.20       2e4          1e-2           0.0          0.0           1.0
0.400       0.020      0.20       2e4          1e-2           0.0          0.0           1.0

```

```matlab
% preProcessing script to prepare input files for LIGGGHTS(R)
% Deformation Code - Fibre Flock Generation
% (c) Lisa Koenig, 2015
%
% Input: Input files in ./input
%        Name: DeformPreProcessing_input_Case_
% Results/ Outputfiles: ./preProcessing
%

%% General
clear all; close all; clc;

%dumpMainStep=10000; % First Array(MainSteps) Last

%% ---------------------- Directories ----------------------------------%
currDir = pwd;                    % current directory
inDir   = '../../input/';         % input directory, contains input files
resDir  = '../../preProcessing/'; % results directory, (LIGGGHTS input)
Visible  = 'off';                 % pop-up figures

%% ------------------ Set and Check Environment ------------------------%
% Check if output folder resDir exists
if exist(resDir) == 0 % if folder does not exist create it!
  mkdir(resDir);
  disp(['Folder: ',fullfile(resDir),' created'])
end

% Check number of input files (cases) and read input files
inFiles=dir(fullfile(inDir,'DeformPreProcessing_input_Case_*.txt'));
nFiles=length(inFiles);                 % number of input files
disp([num2str(nFiles),' input file(s) have been found.'])

% List all Numbers of the input files (number = Name!)
for i=1:1:nFiles
  inFiles(i).numbers=...
      sscanf(inFiles(i).name,' DeformPreProcessing_input_Case_%i.txt');
end

%% ------------------ % Loop over all input files ----------------------%
for icase=1:1:nFiles    % icase NOT (always) equal to Case name!

%% Create folder for each case:
% Case_* (Case Number from input File name number)
cd (resDir)
folderName=['Case_',num2str(inFiles(icase).numbers)];
if exist(folderName) == 0 % if folder does not exist create it!
  mkdir(folderName);
  disp(['Folder: ',fullfile(resDir, folderName),' created']);
end
cd (currDir)

%% Read input files
% incl. calc reduced box dimensions, calc Volume, spherocylinder diameter
[Fib,rhoFluid,DefMod,Box,Vol,LIGGGTHS]=...
    func_ReadInput(inDir,inFiles(icase).name);

%% Calculation of Fibres in each class that fit in reduced final box
% Aspect Ratio of fibres
Fib.AR = Fib.dMajor ./ (Fib.dMinor./Fib.dMinorDefFactor);

% Fibre mass and volume
 % Vol. of all fibres in system
  Vol.Fibtot = Fib.phi * Vol.rfinal;
 % Vol. single fibre in class i; fibre assumed to be ellipsoid
  Vol.Fibi   = Fib.dMajor.^3 ./ Fib.AR .^2 .* pi ./ 6;

  mass.Fibtot = Fib.rho * Vol.Fibtot; % mass of all fibres in system
  mass.Fibi   = Fib.rho * Vol.Fibi;   % mass of one single fibre in class i

% Fluid mass and volume
```

```
 Vol.Fluid = (1-Fib.phi) * Vol.rfinal;
 mass.Fluid   = rhoFluid * Vol.Fluid;

if Fib.r == 0
   % dQr=0 ... number based distribution
   nFibtot = Vol.Fibtot / sum(Fib.dQri .* Vol.Fibi);
   Fib.nFibi = floor(Fib.dQri *nFibtot); % round down to next integer
   Fib.ntot = sum(Fib.nFibi);

elseif Fib.r == 3
   % dQr=3 ... mass based distribution
   Fib.nFibi = floor(Fib.dQri .* mass.Fibtot ./ mass.Fibi);
   Fib.ntot = sum(Fib.nFibi);
else
   error({'basis of fibre distribution not possible, only 0 or 3',...
           '0 ... r=0 (number based), 3 ... r=3 (volume)'})
end

Vol.allFibi = Vol.Fibi.*Fib.nFibi; % Volume of all fibres in each class i
Vol.allFib  = sum(Vol.allFibi);    % Volume of all fibres in system
Fib.dSphere = ((Fib.dMinor.*Fib.dMinor)./(Fib.dMinorDefFactor).^2 ...
              .*Fib.dMajor).^(1/3); %vol.equi. diam.

%% Real phi in reduced final box differs from input due to floor()
Vol.FibtotReal=0;
for Fibi = 1:1:length(Fib.nFibi)
   Vol.FibtotReal = Vol.FibtotReal + Fib.nFibi(Fibi)*Vol.Fibi(Fibi);
end
Fib.phiReal=Vol.FibtotReal/Vol.rfinal;

Fib.massFracReal=(Vol.FibtotReal*Fib.rho)/...
    (Vol.FibtotReal*Fib.rho + (Vol.rfinal-Vol.FibtotReal)*rhoFluid);

%% Consistancy in reduced final (post-deformation) box
Fib.cons = mass.Fibtot / (mass.Fibtot + mass.Fluid);

%% Calculate initial (pre-deformation) box dimension suggestion
% Note: The initial box needs to be big enough to initially position fibre
% via random function without to many overlaps.
Vol.initSeg=sum(Fib.dMajor.^3.*pi./6.*Fib.ntot);

%% Calculate time step suggestion (see: S. Luding-Introduction to DEM)
% Note: This is not the time step used in the simulation. This is just a
% help for choosing the right value! dtDEM from the input file is used!
LIGGGTHS.m12=...             % effective mass
    min(mass.Fibi)*min(mass.Fibi)/(min(mass.Fibi)+min(mass.Fibi));
LIGGGTHS.redDampingNormal=... % rescaled damping coeff.
    Fib.dampingNormal(end)/(2*LIGGGTHS.m12); % rescaled damping coeff.
LIGGGTHS.omega=...           % eigenrequency
    sqrt(Fib.stiffnessNormal(end)/LIGGGTHS.m12 - ...
                LIGGGTHS.redDampingNormal^2);
LIGGGTHS.tc=pi/LIGGGTHS.omega; % characterstic response time / contact time

% dtDEM = 1/50 tc (see Luding)
LIGGGTHS.dtDEMSug=LIGGGTHS.tc/50;       % suggested value, not used!

% dtcompact = 1/20 dtDEM
LIGGGTHS.dtcompactSug=LIGGGTHS.dtDEMSug/20;% suggested value, not used!

%% Generate random fibre position in inital box and write to .txt - File
% Initial Fibre Position: input for LIGGGHTS
FibPos=func_FibPositioning(...
    inFiles(icase).numbers,Fib,Box,DefMod,resDir,folderName);

%% Data LIGGGHTS input Files
LIGGGTHS.epsTot=(1-Box.rfinal(1)/Box.init(1));
LIGGGTHS.epsSingle=LIGGGTHS.koverlap*max(Fib.dMinor)/max(Fib.dMajor);
LIGGGTHS.tRate=log(1-LIGGGTHS.epsSingle)/...
               (LIGGGTHS.compactEvery*LIGGGTHS.dtDEM);
LIGGGTHS.elapsedTime=log(1-LIGGGTHS.epsTot)/LIGGGTHS.tRate;
LIGGGTHS.DEMsteps=LIGGGTHS.elapsedTime/LIGGGTHS.dtDEM;
LIGGGTHS.compactSteps=LIGGGTHS.DEMsteps/LIGGGTHS.compactEvery;
LIGGGTHS.RealDEMsteps=ceil(LIGGGTHS.compactSteps)*LIGGGTHS.compactEvery;

% dumpFrequencyList
% oneb4Last= floor(LIGGGTHS.RealDEMsteps/dumpMainStep)*dumpMainStep
% dumpBetweenList=[dumpMainStep:dumpMainStep:oneb4Last]
% LIGGGTHS.dumpFrequencyList=[1 dumpBetweenList LIGGGTHS.RealDEMsteps]

%% General Output from preProcessing Calculations
outStat=func_OutputData(...
    inFiles(icase).numbers,Box,Fib,Vol,LIGGGTHS,resDir,folderName);

%% Write further LIGGGHTS input files
wStat=func_LIGGGHTSinFiles(inFiles(icase).numbers,mass,Fib,Box,Vol,...
    LIGGGTHS,DefMod,resDir,folderName);

%% Plot Fibre initial position in initial Box
PlotStat=func_PlotinitPos(...
    inFiles(icase).numbers,Box,Fib,FibPos,resDir,folderName,Visible);
disp([['PreProcessing Case ',num2str(inFiles(icase).numbers),' done.']])

end

%% ------------------- End of Program --------- -------------------------%
disp('Preprocessing is done!')
disp('.txt-Files can be found in ./preProcessing/Case_x/*!')
disp('End of program - Have fun...')
```

```
function PlotStat=func_PlotinitPos...
   (caseNbr,Box,Fib,FibPos,resDir,folderName,...
   Visible)

% Plot initial positions of all fibers in channel with square cross section
% Formate for output/plots
fontSizeTitle=20;
fontSizeAxis=14;
fontSizeLabel=20;
markerSize = 9;
lineWidth=2;

% Box Dimensions of inital box
boxX=Box.init(1);
boxY=Box.init(2);
boxZ=Box.init(3);

figName = 'Initial position of all fibers';
scrs = get(0,'screensize');
rect = [10, 10, 1000, 500]; %[left, bottom, width, height]

fig1 = figure('Name',figName,'Position',rect,'Visible',Visible);

subplot(1,2,1);
   for iClass = 1:1:length(Fib.nFibi);
   plot(FibPos(iClass).initialPosYall, FibPos(iClass).initialPosZall,'.');
   hold on;
   end
   axis([-boxY/2 boxY/2 -boxZ/2 boxZ/2]);
   axis square;
   set(gca,'ZLim',[-boxZ/2 boxZ/2]);
   set(gca,'YLim',[-boxY/2 boxY/2]);
   title('initial fiber distribution y-z',...
       'Interpreter','latex','fontsize',fontSizeTitle);
   xlabel('y','Interpreter','latex');
   ylabel('z','Interpreter','latex');

   grid off
   box on
   legend boxoff
   set(0,'defaultaxesfontsize',fontSizeAxis);
   set(0,'defaulttextfontsize',fontSizeAxis);
   set(gca,'FontSize',fontSizeAxis);
   set(gca,'FontWeight','normal');
   xlhand = get(gca,'xlabel');
   ylhand = get(gca,'ylabel');
   zlhand = get(gca,'zlabel');
   set(xlhand,'fontsize',fontSizeLabel);
   set(ylhand,'fontsize',fontSizeLabel);
   set(zlhand,'fontsize',fontSizeLabel)
   set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
   set(xlhand,'FontWeight','bold');
   set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
   set(ylhand,'FontWeight','bold');
   set(zlhand,'FontName','Times'); set(zlhand,'FontAngle','italic');
   set(zlhand,'FontWeight','bold');

subplot(1,2,2)
   for iClass = 1:1:length(Fib.nFibi)
   plot(FibPos(iClass).initialPosXall, FibPos(iClass).initialPosZall,'.');
   hold on;
   end
   axis([-boxX/2 boxX/2 -boxZ/2 boxZ/2]);
   axis equal;
   set(gca,'XLim',[-boxX/2 boxX/2]);
   set(gca,'YLim',[-boxZ/2 boxZ/2]);
   title('initial fiber distribution x-z',...
       'Interpreter','latex','fontsize',fontSizeTitle);
   xlabel('x','Interpreter','latex');
   ylabel('z','Interpreter','latex');

   grid off
   box on
   legend boxoff
   set(0,'defaultaxesfontsize',fontSizeAxis);
   set(0,'defaulttextfontsize',fontSizeAxis);
   set(gca,'FontSize',fontSizeAxis);
   set(gca,'FontWeight','normal');
   xlhand = get(gca,'xlabel');
   ylhand = get(gca,'ylabel');
   zlhand = get(gca,'zlabel');
   set(xlhand,'fontsize',fontSizeLabel);
   set(ylhand,'fontsize',fontSizeLabel);
   set(zlhand,'fontsize',fontSizeLabel)
   set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
   set(xlhand,'FontWeight','bold');
   set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
   set(ylhand,'FontWeight','bold');
   set(zlhand,'FontName','Times'); set(zlhand,'FontAngle','italic');
   set(zlhand,'FontWeight','bold');

% Print pdf
print('-dpng','-r300',...
   [resDir,folderName,'/FibrePositionInitBox_Case_',num2str(caseNbr)]);

PlotStat=1;

end

```

```matlab
function [Fib,rhoFluid,DefMod,Box,Vol,LIGGGHTS]=...
    func_ReadInput(inDir,fileName)
% Read input files and calc reduced final box dimensions

% Get Data from input File via dlmrad()
temp.Data1 = dlmread(strcat(inDir,fileName),'',[28 0 28 3]);
temp.Data2 = dlmread(strcat(inDir,fileName),'',[31 0 31 2]);
temp.Data3 = dlmread(strcat(inDir,fileName),'',[36 0 36 2]);
temp.Data4 = dlmread(strcat(inDir,fileName),'',[39 0 39 2]);
temp.Data5 = dlmread(strcat(inDir,fileName),'',[44 0 44 4]);
temp.Data6 = dlmread(strcat(inDir,fileName),'',47,0);

%% DEM PARAMETERS
LIGGGHTS.roughFact=temp.Data1(1);
LIGGGHTS.roughStiffFact=temp.Data1(2);
LIGGGHTS.frictionCoeff=temp.Data1(3);
LIGGGHTS.liquidDynViscosity=temp.Data1(4);

LIGGGHTS.dtDEM = temp.Data2(1);
LIGGGHTS.koverlap = temp.Data2(2);
LIGGGHTS.compactEvery = temp.Data2(3);

%% BOX DIMENSIONS
% Initial (pre-deformation) Box Dimensions
Box.init = temp.Data3;

% Final (post-deformation) Box Dimensions
Box.final = temp.Data4;

%% FIBRE/FLUID DATA & DEFORMATION MODE:
Fib.r    = temp.Data5(1);
Fib.rho  = temp.Data5(2);
rhoFluid = temp.Data5(3);
Fib.phi  = temp.Data5(4);
DefMod   = temp.Data5(5);

% Fibre Distribution Data
Fib.Data = sortrows(temp.Data6,[1 2]); % Sort acc. to dMajor
Fib.dMajor = Fib.Data(:,1);
Fib.dMinor = Fib.Data(:,2);
Fib.dQri   = Fib.Data(:,3);
Fib.stiffnessNormal   = Fib.Data(:,4);
Fib.dampingNormal   = Fib.Data(:,5);
Fib.stiffnessTang   = Fib.Data(:,6);
Fib.dampingTang   = Fib.Data(:,7);
Fib.dMinorDefFactor   = Fib.Data(:,8); % run deformation with different dMinor

% reduced initial (pre-deformation) box
redValue=max(Fib.dMajor);
Box.rinit = Box.init -[redValue redValue redValue];

% reduced final (post-deformation) box
if DefMod == 1;     % uni-aixal deformation in x-direction
  Box.rfinal = Box.final -[redValue redValue redValue];
elseif DefMod == 3; % tri-aixal deformation in all directions
  Box.rfinal = Box.final -[redValue redValue redValue];
end

% Box Volume
% reduced final(post-deformation) box for fibres after deformation
Vol.rfinal = Box.rfinal(1)* Box.rfinal(2)* Box.rfinal(3);

% diameter of spherocylinder
Fib.dCylinder=1.24.*Fib.dMinor./sqrt(log(Fib.dMajor./Fib.dMinor));

clear temp.* % clear temporary data
end

```

```matlab
function FibPos=func_FibPositioning(caseNbr,Fib,Box,DefMod,...
  resDir,folderName)
% Create LIGGGHTS input file holding the fibres and ther position:
% create_atoms    1 single initialPosX initialPosY1 initialPosZ1 units box

% Box Dimensions for fibre positions
% Note: periodic boundary condition in LIGGGHTS code deform

if DefMod == 1; % uni-axial in x.direction
 boxX=Box.init(1);  % deformation direction: x
 boxY=Box.rinit(2); % red. already at beginning due to no deform in y-dir
 boxZ=Box.rinit(3); % red. already at beginning due to no deform in z-dir
elseif DefMod == 3; % tri-axial equally in all directions
 boxX=Box.init(1); % deformation direction: x
 boxY=Box.init(2); % deformation direction: y
 boxZ=Box.init(3); % deformation direction: z
end

% Include File
includeFile=fopen(fullfile(resDir,folderName,['in.AllParticlePos_Case_',...
      num2str(caseNbr),'.txt']), 'w');

rand('seed',666)  %  MATLAB rng(666) does not work in Octave!
 % initialize random function: predictable sequence of numbers
for nFibClass=1:1:length(Fib.nFibi)
  atom_type=nFibClass;
  outFile=fopen(fullfile(resDir,folderName,['in.ParticlePos_Case_',...
      num2str(caseNbr),'_AR',num2str(Fib.AR(nFibClass)),'.txt']), 'w');

  % Generate fiber initial positions
  % Fibers: X-direction
  InterX = [-boxX/2 boxX/2];
  FibPos(nFibClass).initialPosXall = ...
     InterX(1) + (InterX(2)-InterX(1)).*rand(Fib.nFibi(nFibClass),1);

  % Fibers: Y-direction
  InterY = [-boxY/2 boxY/2];
  FibPos(nFibClass).initialPosYall = ...
     InterY(1) + (InterY(2)-InterY(1)).*rand(Fib.nFibi(nFibClass),1);

  % Fibers: Z-direction
  InterZ = [-boxZ/2 boxZ/2];
  FibPos(nFibClass).initialPosZall = ...
     InterZ(1) + (InterZ(2)-InterZ(1)).*rand(Fib.nFibi(nFibClass),1);

  % Create array with Initial fiber Position
  FibPos(nFibClass).initialPos =...
     [FibPos(nFibClass).initialPosXall...
      FibPos(nFibClass).initialPosYall...
      FibPos(nFibClass).initialPosZall];

  % Write LIGGGHTS input files: Init. fibre Position: in.ParticlePos_Case_
   for n = 1:1:Fib.nFibi(nFibClass)
        fprintf( outFile, '%s %i %s %5.4f %5.4f %5.4f %s \n',...
        'create_atoms    ',atom_type, 'single ',...
         FibPos(nFibClass).initialPos(n,1),...
         FibPos(nFibClass).initialPos(n,2),...
         FibPos(nFibClass).initialPos(n,3),...
        ' units box ');
   end

   fprintf(includeFile, '%s %s \n','include ',...
     ['./preProcessing/',folderName,['/in.ParticlePos_Case_',...
     num2str(caseNbr),...
     '_AR',num2str(Fib.AR(nFibClass)),'.txt']]);

end



```

```
function wStat=func_LIGGGHTSinFiles(caseNbr,mass,Fib,Box,Vol,LIGGGTHS,...
   DefMod,resDir,folderName)

%% Write further LIGGGHTS input files

%% DEM INPUT
% =========

% Interaction_Parameter
File1=fopen(fullfile(resDir,folderName,...
    ['in.DEM_Input_Case_',num2str(caseNbr),'.txt']), 'w');

fprintf(File1,'%s \n',      '# USER Input');
fprintf(File1,'%s %12.10f %s\n','variable timestepValue       equal ',...
        LIGGGTHS.dtDEM,' #dtDEM');
fprintf(File1,'%s \n',      '');

fprintf(File1,'%s %10.8f %s, \n','variable rhoParticle equal ',...
             Fib.rho,'#particle density');
fprintf(File1,'%s %3i %s, \n',   'variable nAtomTypes  equal ',...
             length(Fib.AR),'# number of atom types');
fprintf(File1,'%s %10.8f %s, \n',   'variable FibVol      equal ',...
            Vol.allFib,'# Volume of all fibres in all classes');
fprintf(File1,'%s \n','');

fprintf(File1,'%s \n','# Fibre Types: ');
for ii=1:1:length(Fib.AR);
fprintf(File1,'%s %10.8f %s \n',...
    ['variable AR0'      ,num2str(ii),'             equal '],...
    Fib.AR(ii),...
    '#major diameter of ellipsoid (=total length of spherocylinder)');
fprintf(File1,'%s %10.8f %s \n',...
    ['variable dMajorAR0'      ,num2str(ii),'       equal '],...
    Fib.dMajor(ii),...
    '#major diameter of ellipsoid (=total length of spherocylinder)');
fprintf(File1,'%s %10.8f %s \n',...
    ['variable dMinorAR0'      ,num2str(ii),'       equal '],...
    Fib.dMinor(ii),...
    '#minor diameter of ellipsoid (NOT that of spherocylinder)');
fprintf(File1,'%s %10.8f %s \n',...
    ['variable dCylinderAR0'   ,num2str(ii),'    equal '],...
    Fib.dCylinder(ii),'#diameter of spherocylinder');
fprintf(File1,'%s %10.8f %s \n',...
    ['variable massParticleAR0',num2str(ii),' equal '],...
    mass.Fibi(ii),'#mass of particle');
fprintf(File1,'%s %10.8f %s \n',...
    ['variable volParticleAR0' ,num2str(ii),'  equal '],...
    Vol.Fibi(ii),'#volume of particle');
fprintf(File1,'%s %10.8f %s \n',...
    ['variable dSphereAR0'     ,num2str(ii),'      equal '],...
    Fib.dSphere(ii),'#equivalent diameter');
fprintf(File1,'%s \n','');
end

fprintf(File1,'%s \n','');

fprintf(File1,'%s \n',      '# Interaction Parameter for force Calc ');
for ii=1:1:length(Fib.AR)
fprintf(File1,'%s %10.8f\n',...
    ['variable stiffnessNormalAR0',num2str(ii),'     equal '],...
    Fib.stiffnessNormal(ii));
fprintf(File1,'%s %10.8f\n',...
    ['variable dampingNormalAR0',num2str(ii),'       equal '],...
    Fib.dampingNormal(ii));
fprintf(File1,'%s %10.8f\n',...
    ['variable stiffnessTangAR0',num2str(ii),'       equal '],...
    Fib.stiffnessTang(ii));
fprintf(File1,'%s %10.8f\n',...
    ['variable dampingTangAR0',num2str(ii),'         equal '],...
    Fib.dampingTang(ii));
fprintf(File1,'%s \n','');
end

fprintf(File1,'%s %10.8f\n','variable roughFact           equal ',...
    LIGGGTHS.roughFact);
fprintf(File1,'%s %10.8f\n','variable roughStiffFact      equal ',...
    LIGGGTHS.roughStiffFact);
fprintf(File1,'%s %10.8f\n','variable frictionCoeff       equal ',...
    LIGGGTHS.frictionCoeff);
fprintf(File1,'%s %10.8f\n','variable liquidDynViscosity  equal ',...
    LIGGGTHS.liquidDynViscosity);
fprintf(File1,'%s \n',      '');

for ii=1:1:length(Fib.AR)
fprintf(File1,'%s \n',...
    ['variable roughnessAR0'             ,num2str(ii),...
    '    equal ${roughFact}*${dMinorAR0',num2str(ii),'} #roughness']);
fprintf(File1,'%s \n',...
    ['print     ''roughnessAR0',num2str(ii),...
    ': ${roughnessAR0',num2str(ii),'}''']);
end

fprintf(File1,'%s \n',      '');
fprintf(File1,'%s \n','# contact times');
for ii=1:1:length(Fib.AR)
fprintf(File1,'%s \n',...
    ['variable tContactAR0',num2str(ii),...
    '  equal PI/sqrt(${stiffnessNormalAR0',num2str(ii),...
    '}/(${massParticleAR0',num2str(ii),...
    '}*0.5)-${dampingNormalAR0',num2str(ii),'}^2)']);
end
fprintf(File1,'%s \n',      '');

% BOX Dimenions initial
fprintf(File1,'%s \n',      '# Box dimensions');
fprintf(File1,'%s \n',...
    '# initial (pre-deformation) box dimensions = LIGGGTHS Simulation box');
if DefMod == 1; % uni-axial in x.direction
    fprintf(File1,'%s %10.8f\n',...
        'variable sizeDomainInitHalfX     equal ',Box.init(1)/2);
    fprintf(File1,'%s %10.8f\n',...
        'variable sizeDomainInitHalfY     equal ',Box.rinit(2)/2);
    fprintf(File1,'%s %10.8f\n',...
        'variable sizeDomainInitHalfZ     equal ',Box.rinit(3)/2);
elseif DefMod == 3; % tri-axial equally in all directions
    fprintf(File1,'%s %10.8f\n',...
        'variable sizeDomainInitHalfX     equal ',Box.init(1)/2);
    fprintf(File1,'%s %10.8f\n',...
        'variable sizeDomainInitHalfY     equal ',Box.init(2)/2);
    fprintf(File1,'%s %10.8f\n',...
        'variable sizeDomainInitHalfZ     equal ',Box.init(3)/2);
end
fprintf(File1,'%s \n',      '');
fprintf(File1,'%s \n',...
    '# reduced final (post-deformation) box dimensions');
fprintf(File1,'%s %10.8f\n',...
    'variable sizeDomainFinalHalfX     equal ',Box.rfinal(1)/2);
fprintf(File1,'%s %10.8f\n',...
    'variable sizeDomainFinalHalfY     equal ',Box.rfinal(2)/2);
fprintf(File1,'%s %10.8f\n',...
    'variable sizeDomainFinalHalfZ     equal ',Box.rfinal(3)/2);
fprintf(File1,'%s \n',      '');

% Deform Box Parameter
fprintf(File1,'%s \n','# Parameter for deform Box command');
fprintf(File1,'%s %10.8f %s\n',...
    'variable compactEveryNthTimeStep equal ',...
    LIGGGTHS.compactEvery,' # compact every this many time steps');
fprintf(File1,'%s %10.8f %s\n',...
    'variable tRateValue              equal ',...
```

```matlab
    LIGGGTHS.tRate,' # tRate');
fprintf(File1,'%s %10.8f %s\n',...
    'variable RealDEMsteps            equal ',...
    LIGGGTHS.RealDEMsteps,...
' # time steps needed to garantee red. final box dim. at end of simu');
fprintf(File1,'%s \n',       '');


%% in.GroupFibres_Case_
% =====================
File2=fopen(fullfile(resDir,folderName,...
    ['in.GroupFibres_Case_',num2str(caseNbr),'.txt']), 'w');
fprintf(File2,'%s \n','# Group Fibres acc. to AR');
fprintf(File2,'%s %10.8f %s\n',...
    'variable compactEveryNthTimeStep equal ',...
    LIGGGTHS.compactEvery,' # compact every this many time steps');

for ii=1:1:length(Fib.AR)

if gt(Fib.nFibi(ii),0)
fprintf(File2,'%s \n',['# Group: ',num2str(ii)]);
fprintf(File2,' %s %s \n',['set   type ',num2str(ii),...
    ' diameter ${dSphereAR0',num2str(ii),'} density ${rhoParticle} '],...
    '#major diameter of ellipsoid (=total length of spherocylinder)');
fprintf(File2,' %s %s \n',['set   type ',num2str(ii),...
    ' mass ${massParticleAR0',num2str(ii),'} '],'# mass of the particle)');
fprintf(File2,' %s %s \n',['set   type ',num2str(ii),...
    ' shape ${dMajorAR0',num2str(ii),'} ${dMinorAR0',num2str(ii),...
    '} ${dMinorAR0',num2str(ii),'}'],'# shape)');
fprintf(File2,'%s \n',['group groupAR0',num2str(ii),...
    ' type ',num2str(ii)]);
fprintf(File2,'%s \n',['variable nFibAR0',num2str(ii),...
    ' equal count(groupAR0',num2str(ii),') # number of fibres in group']);
fprintf(File2,'%s \n','');
end

end

% Deform Line depends deponds if uni- or tri-axial deformation is used
File3=fopen(fullfile(resDir,folderName,...
    ['in.DeformFix_Case_',num2str(caseNbr),'.txt']), 'w');

if DefMod == 1;
fprintf(File3,'%s %s\n',...
'fix    ',...
'defBox all deform ${compactEveryNthTimeStep} x trate ${tRateValue}');
elseif DefMod == 3;
fprintf(File3,'%s %s\n',...
'fix     defBox all deform ${compactEveryNthTimeStep} x',...
'trate ${tRateValue} y trate ${tRateValue} z trate ${tRateValue}');
end

wStat=1;
end

```

```matlab
function outStat=func_OutputData(CaseNbr,Box,Fib,Vol,LIGGGTHS,...
    resDir,folderName)
% Function to write output Data in .txt-file
outFile=fopen(fullfile(resDir,folderName,['Data_Case_',num2str(CaseNbr),...
                        '.txt']), 'w');
fprintf(outFile,'\n %s \n',' Case Data ');
fprintf(outFile,'%s \n','===================');
fprintf(outFile,'%s %6.3f %s %6.3f %s %6.3f \n',...
 'org. Box       = ',Box.init(1),' x ',Box.init(2),' x ',Box.init(3));
fprintf(outFile,'%s %6.3f %s %6.3f %s %6.3f \n',...
  'red. def. Box = ',Box.rfinal(1),' x ',Box.rfinal(2),' x ',Box.rfinal(3));
fprintf(outFile,'%s \n'    ,'');
fprintf(outFile,'%s %4.4e \n'    ,...
    'Fibre Volume Fraction:      VFibtot / Vred.def. = ',Fib.phi);
fprintf(outFile,'%s %4.4e \n'    ,...
    'Real Fibre Volume Fraction: VFibtot / Vred.def.   = ',Fib.phiReal);
fprintf(outFile,'%s %4.4e \n'    ,'Real Mass Frac  = ',Fib.massFracReal);
fprintf(outFile,'%s %4.6e \n'    ,'Consistency = ',Fib.cons);
fprintf(outFile,'%s %6i   \n'    ,'#tot fibres = ',Fib.ntot);
fprintf(outFile,'%s %4.6e \n'    ,'Tot Vol Fibres = ',Vol.Fibtot);
fprintf(outFile,'%s %4.6e \n'    ,'Tot Vol Fibres Real= ',Vol.FibtotReal);
fprintf(outFile,'%s %4.6e \n'    ,'Vreddeftot  = ',Vol.rfinal);
fprintf(outFile,'%s \n'    ,'');
fprintf(outFile,'%s \n'    ,'Suggested Values:');
fprintf(outFile,'%s %14.10e \n'  ,'suggested dtDEM  = ',LIGGGTHS.dtDEMSug);
fprintf(outFile,'%s %14.10e \n'  ,'suggested initial Vol  = ',Vol.initSeg);

fprintf(outFile,'%s \n'    ,'');
fprintf(outFile,'%s \n'    ,'Fibres in each class i');
fprintf(outFile,'%s \n'    ,'=====================');
for j=1:1:length(Fib.nFibi)
    fprintf(outFile,'%s \n','---');
    fprintf(outFile,'%s %i %s %4.6f \n','dMajor(',j,') = ',Fib.dMajor(j));
    fprintf(outFile,'%s %i %s %4.6f \n','dMinor(',j,') = ',Fib.dMinor(j));
    fprintf(outFile,'%s %i %s %4.6f \n','AR(',j,')      = ',Fib.AR(j));
    fprintf(outFile,'%s %i %s %4.6f \n','dQri(',j,')    = ',Fib.dQri(j));
    fprintf(outFile,'%s %i %s %4.6f \n','nFibi(',j,')  = ',Fib.nFibi(j)) ;
end

outStat=1;
end
```

```matlab
% PostProcessing script for LIGGGTHS
% Deformation Code - Fibre Flock Generation
% (c) Lisa Koenig, 2015
% Only fibre Orientation at beginning and end of simulation
% NO angular velocity in Fibre Flock generation!
% clear all; close all; clc

% x   y   z    ... global coordinate system (x, y, z)
% x'  y'  z'  ... coordinate system in fibres' center (xS, yS, zS)
% x'' y'' z'' ... coordinate system in fibre direction (xSS, ySS, zSS)
%
% Input files: dump files from LIGGGHTS(R)
% dump-file: atom_data must have this form:
% id type x y z vx vy vz fx fy fz
% 1   2   3 4 5 6  7  8  9  10 11
%
% omegax omegay omegaz radius f_ex[1] f_ex[2] f_ex[3] f_shape[1]
%   12     13     14     15     16      17      18      19
%
clear all; close all; clc;

%% --------------------------- Global Parameters ------------------------ %
global homeDir resDirMain resDirSub inDir FigVisible interpreterName ...
printAs dMinor

FigVisible='on';           % Visible Figures 'on' or 'off'
printAs='-dpng';            % '-dpdf' (PDF), '-dpng' (PNG)

dMinor=0.02;

% Input from tempData (bashScript)
%interpreterName='latex';    % 'latex' (MATLAB), 'tex' (Octave/GNUplot)
%caseIdToRun=1;

tempDataFile = fopen('tempData.txt');
C = textscan(tempDataFile,'%f %f',1);
fclose(tempDataFile);
caseIdToRun=C{1,1}
InterPreterName=C{1,2}

if InterPreterName == 1 % MATLAB
    interpreterName='latex'
end

if InterPreterName == 2 %OCTAVE
    interpreterName='tex'
end

%% --------------------------- Directories ------------------------------ %
% home/main direcoty
homeDir = pwd;

 % directory for saving calc data, txt-files...
resDirMain  = '../../postProcessing/';

% directory for saving figures, pdfs,... of that Case
resDirSub   = ['./Case_',num2str(caseIdToRun),'/'];

% directory of INput files (dump-files)
inDir       = ['../../post_Case_',num2str(caseIdToRun),'/'];

% Check if output folder ./resDirMain exists
if exist(resDirMain) == 0 % if folder does not exist create it!
   mkdir(resDirMain);
   disp({['Folder: ',fullfile(resDirMain),' created']})
end

% Check if output folder ./resDirMain/resDirSub exists
if exist(fullfile(resDirMain,resDirSub)) == 0 % doesn't exist,create it!
   mkdir(fullfile(resDirMain,resDirSub));
   disp({['Folder: ',fullfile(resDirMain,resDirSub),' created']})
end

%% Check number of dump-files
cd(inDir)                        % fo in directory with dump-Files
files = dir('dump*liggghts');   % list all dump-files
cd(homeDir)                      % go back to home directory
nDmpFiles=length(files);        % number of dump-files in folder

% Read all dump-files
for dmpNbr=1:1:nDmpFiles
   dump(dmpNbr) = readdump_all(fullfile(inDir,files(dmpNbr).name));
end

% Loop over all dump files
for dmpNbr=1:1:nDmpFiles
   % Sort dump-file acc. to atom ID
   dump(dmpNbr).atom_data = sortrows(dump(dmpNbr).atom_data,1);

   % Divide Fibres dump - file acc. to their position
   dump(dmpNbr).Section = func_Section(dump(dmpNbr));

   % Coordinate Transformation (90 => pi/2) acc. to Section 1-4
   % =========================================================
   % GLOBAL SYSTEM TO SECTION SYSTEM x -> x''
   % Rotation with general rotation matrix D
   % Section 1: no transformation needed
   % Section 2: transform.  -Pi around x-axis, basis global coord. sys.
   %                        orientated like Section 1 coordinate sys.
    % Section 3: transform. -2 Pi around x-axis, -||-
    % Section 4: transform. -3 Pi around x-axis, -||-

   for i=1:1:4 %4 Sections
      % rotation angle in rad
      dump(dmpNbr).Section(i).rotAngleG2SS=pi/2 .*(i-1);
      % rotation axis -> x-axis
      dump(dmpNbr).Section(i).rotAxisG2SS=[1 0 0];
      % rotation matrix
      D=func_GeneralRotationMatrix(dump(dmpNbr).Section(i).rotAxisG2SS,...
                                   dump(dmpNbr).Section(i).rotAngleG2SS);
     % ID and Type => 1:2 in dump
      dump(dmpNbr).Section(i).atom_data_SS(:,1:2)=...
          dump(dmpNbr).Section(i).atom_data_G(:,1:2);

     % x y z => 3:5 in dump
      TempVec2Rot=dump(dmpNbr).Section(i).atom_data_G(:,3:5).';
      dump(dmpNbr).Section(i).atom_data_SS(:,3:5)=...
          (D*TempVec2Rot).';

     % Velocity Vector vx vy vz => 6:8 in dump
      TempVec2Rot=dump(dmpNbr).Section(i).atom_data_G(:,6:8).';
      dump(dmpNbr).Section(i).atom_data_SS(:,6:8)=...
          (D*TempVec2Rot).';

     % Force Vector fx fy fz => 9:11 in dump
      TempVec2Rot=dump(dmpNbr).Section(i).atom_data_G(:,9:11).';
      dump(dmpNbr).Section(i).atom_data_SS(:,9:11)=...
          (D*TempVec2Rot).';

     % Angle Velocity Vector omegax omegay omegaz => 12:14 in dump
      TempVec2Rot=dump(dmpNbr).Section(i).atom_data_G(:,12:14).';
      dump(dmpNbr).Section(i).atom_data_SS(:,12:14)=...
          (D*TempVec2Rot).';

     % Radius -> does not change -> 15 in dump
      dump(dmpNbr).Section(i).atom_data_SS(:,15)=...
          dump(dmpNbr).Section(i).atom_data_G(:,15);

     % Orientation Vector f_ex[1] f_ex[2] f_ex[3] => 16:18 in dump
      TempVec2Rot=dump(dmpNbr).Section(i).atom_data_G(:,16:18).';
```

```
        dump(dmpNbr).Section(i).atom_data_SS(:,16:18)=...
            (D*TempVec2Rot).';

       % Shape -> does not change -> 19 in dump
        dump(dmpNbr).Section(i).atom_data_SS(:,19)=...
            dump(dmpNbr).Section(i).atom_data_G(:,19);
   end
end

%% Fibre Orientation Analysis - Compare First and Last Dump-File
% Check for first and last timeStep to get first and last dump-file
disp('Starting Fibre Orientation Analysis: First vs. Last Dump-File')
aTemp=[dump(:).timestep];
FirstDump = min(aTemp(aTemp>0))
IndFirstDump=min(find(aTemp))
% [FirstDump,IndFirstDump] = min(aTemp(aTemp>0));
[LastDump,IndLastDump]    = max([dump(:).timestep])
FLIndex=[IndFirstDump, IndLastDump]; % First Last Index

ResultsFolder=fullfile(resDirMain,resDirSub);

for i =1:1:2 %First and last dump File
dmpNbr = FLIndex(i);
% Calc. Wall-distance (always z-direction!)
RefzDist=abs(dump(dmpNbr).z_bound(1));
      dump(dmpNbr).Section(1).FibOrient.zDist(:,1)= ...
          (abs(dump(dmpNbr).Section(1).atom_data_SS(:,5)))/RefzDist;

      dump(dmpNbr).Section(2).FibOrient.zDist(:,1)= ...
          (abs(dump(dmpNbr).Section(2).atom_data_SS(:,5)))/RefzDist;

      dump(dmpNbr).Section(3).FibOrient.zDist(:,1)= ...
          (abs(dump(dmpNbr).Section(3).atom_data_SS(:,5)))/RefzDist;

      dump(dmpNbr).Section(4).FibOrient.zDist(:,1)= ...
          (abs(dump(dmpNbr).Section(4).atom_data_SS(:,5)))/RefzDist;

% Fibre Orientation Analysis
% Calc phi and theta
%  theta=atan(y''/x'') x'',y'',z'' of orientation vector f_ex
%  phi=acos(z'')

    dump(dmpNbr).Section(i).FibOrient.theta=[];
    dump(dmpNbr).Section(i).FibOrient.phi=[];
    dump(dmpNbr).Section(i).FibOrient.radius=[];

    dump(dmpNbr).FibOrient.theta=[];
    dump(dmpNbr).FibOrient.phi=[];
    dump(dmpNbr).FibOrient.radius=[];

for i=1:1:4 % 4 sections

  for j=1:1:size(dump(dmpNbr).Section(i).atom_data_SS,1);
   dump(dmpNbr).Section(i).FibOrient.theta(j,1)= ...
       dump(dmpNbr).Section(i).atom_data_SS(j,19); % shape
   dump(dmpNbr).Section(i).FibOrient.phi(j,1)  = ...
       dump(dmpNbr).Section(i).atom_data_SS(j,19); % shape

   [dump(dmpNbr).Section(i).FibOrient.theta(j,2),...
    dump(dmpNbr).Section(i).FibOrient.phi(j,2)] = ...
       func_ConvCart2Polar(dump(dmpNbr).Section(i).atom_data_SS(j,16:18));
       % in rad

  end
  % Sum all thetas, phis and zDist in one array for each dump-file
   dump(dmpNbr).FibOrient.theta=vertcat(dump(dmpNbr).FibOrient.theta,...
       dump(dmpNbr).Section(i).FibOrient.theta(:,:));
   dump(dmpNbr).FibOrient.phi  =vertcat(dump(dmpNbr).FibOrient.phi,...
       dump(dmpNbr).Section(i).FibOrient.phi(:,:));
   dump(dmpNbr).FibOrient.radius=vertcat(dump(dmpNbr).FibOrient.radius,...
       dump(dmpNbr).Section(i).FibOrient.zDist(:,:));
end

 %% Angle Distribution - Histrogramms
%  [dump(dmpNbr).HistAllAzi,dump(dmpNbr).HistAllPol]=...
%      func_PlotHist(dump(dmpNbr),...
%      dump(dmpNbr).FibOrient.theta,...
%      dump(dmpNbr).FibOrient.phi,ResultsFolder);

 %% Angle Distribution - with Fibre Length
[dump(dmpNbr).HistAziBins,dump(dmpNbr).HistAzi,dump(dmpNbr).HistAziNorm,...
 dump(dmpNbr).HistPolBins,dump(dmpNbr).HistPol,dump(dmpNbr).HistPolNorm]...
    = func_PlotHist_FibClass(dump(dmpNbr),dump(dmpNbr).FibOrient.theta,...
                         dump(dmpNbr).FibOrient.phi,...
                         ResultsFolder);

%  FigVisible='on';
%% Angle Distibution in polar plot - with Fibre Length Distribution
% [dump(dmpNbr).HistAziBins,dump(dmpNbr).HistAzi,dump(dmpNbr).HistAziNorm,...
%  dump(dmpNbr).HistPolBins,dump(dmpNbr).HistPol,dump(dmpNbr).HistPolNorm]...
% = func_PlotHistPolar_FibClass(dump(dmpNbr),dump(dmpNbr).FibOrient.theta,...
%                          dump(dmpNbr).FibOrient.phi,...
%                          ResultsFolder);

%% -------------------------- Plot Figures ---------------------------- %
% Theta ... azimuthal angle & Phi ... polar angle
% PlotStatus=func_PlotAngle(dump(dmpNbr),ResultsFolder);
end

disp('End of program... Have fun!')
```

```
function [varargout] = readdump_all(varargin)
% Reads all timesteps from a LAMMPS dump file.
% Input is dump file name with path
% Output is in the form of a structure with following variables
% .timestep     --> Vector containing all time steps
% .Natoms       --> Vector containing number of atoms at each time step
% .x_bound      --> [t,2] array with xlo,xhi at each time step
% .y_bound      --> [t,2] array with ylo,yhi at each time step
% .z_bound      --> [t,2] array with zlo,zhi at each time step
% .atom_data    --> 3 dimensional array with data at each time step stored
%                   as atomdata(:,:,t)
% Example
%       data = readdump_all('dump.LAMMPS');
%
% See also readdump_one, scandump
%
%  Author :  Arun K. Subramaniyan
%           sarunkarthi@gmail.com
%           http://web.ics.purdue.edu/~asubrama/pages/Research_Main.htm
%           School of Aeronautics and Astronautics
%           Purdue University, West Lafayette, IN - 47907, USA.

try
    dump = fopen(varargin{1},'r');
catch
    error('Dumpfile not found!');
end

i=1;
while feof(dump) == 0
    id = fgetl(dump);
     if (strcmpi(id,'ITEM: TIMESTEP'))
            timestep(i) = str2num(fgetl(dump));
    else
     if (strcmpi(id,'ITEM: NUMBER OF ATOMS'))
            Natoms(i) = str2num(fgetl(dump));
     else
      if (strncmpi(id,'ITEM: BOX BOUNDS',15))
            x_bound(i,:) = str2num(fgetl(dump));
            y_bound(i,:) = str2num(fgetl(dump));
            z_bound(i,:) = str2num(fgetl(dump));
      else
       if (strncmpi(id,'ITEM: ATOMS',10))
            for j = 1 : 1: Natoms
                atom_data(j,:,i) = str2num(fgetl(dump));
            end
            i=i+1;
       end
      end
     end
    end
end
%----------Outputs-------------
%OUTPUTS IN SAME VARIABLE STRUCTURE
varargout{1}.timestep = timestep;
varargout{1}.Natoms = Natoms;
varargout{1}.x_bound = x_bound;
varargout{1}.y_bound = y_bound;
varargout{1}.z_bound = z_bound;
varargout{1}.atom_data = atom_data;
%------------------------------
fclose(dump);
```

```
%% Figure - plot angle (theta, or phi)
function PlotStatus=func_PlotAngle(dump,ResultsFolder)
global FigVisible interpreterName printAs

%% Get Screen Size
%scrsz = get(groot,'ScreenSize');% MATLAB >R2014b
scrsz = get(0,'ScreenSize'); % MATLAB <R2014b

%% --------------------------- Formating ------------------------------ %
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

%% -------------------- AZIMUTHAL ANGLE -------------------------------- %
    Name=['HistPlotPolar_Azi_dump',num2str(dump.timestep)];
    % Create Figrue
    fig1=figure('Name',Name,'Position',...
        [1 scrsz(4)/2.2 scrsz(3)/2.5 scrsz(4)/2],'Visible',FigVisible);
    h1=polar(dump.Section(1).FibOrient.theta(:,2),...
        dump.Section(1).FibOrient.zDist(:,1),...
        '^');hold on;
        set(h1,'Color',[1 0.5 0])
    h2=polar(dump.Section(2).FibOrient.theta(:,2),...
        dump.Section(2).FibOrient.zDist(:,1),...
        'o');hold on;
        set(h2,'Color',[0 0 1])
    h3=polar(dump.Section(3).FibOrient.theta(:,2),...
        dump.Section(3).FibOrient.zDist(:,1),...
        's');hold on;
        set(h3,'Color',[1 0 0])
    h4=polar(dump.Section(4).FibOrient.theta(:,2),...
        dump.Section(4).FibOrient.zDist(:,1),...
        'd');hold off;
        set(h4,'Color',[0 0 0])

legendString={[preAfterSymbol,'1^{st} section',preAfterSymbol],...
      [preAfterSymbol,'2^{nd} section',preAfterSymbol],...
      [preAfterSymbol,'3^{rd} section',preAfterSymbol],...
      [preAfterSymbol,'4^{th} section',preAfterSymbol]};

legend(legendString,'Location','NorthEastOutside','Orientation','vertical')

 if(  strcmp(interpreterName, 'latex') )
  legend(legendString,...
       'Location','NorthEastOutside','Orientation','vertical',...
       'interpreter',interpreterName,'FontSize',labelFontSize)
 end

    title([preAfterSymbol,'\theta',preAfterSymbol,...
          ' - azimuthal angle'],...
          'interpreter',interpreterName,'FontSize',fontSizeTitle)
    grid off;
    box on;
    set(0,'defaultaxesfontsize',14);

    xlabel([preAfterSymbol,'r^{+}',preAfterSymbol,...
           ' : distance from the centre ',...
           'with ',preAfterSymbol,'r^{+}',preAfterSymbol,...
            ' = 1 at the wall'],...
    'interpreter',interpreterName,'fontsize',labelFontSize)

    set(0,'defaultaxesfontsize',stdTextFontSize);
    set(0,'defaulttextfontsize',stdTextFontSize);
    set(gca,'FontSize',stdTextFontSize);
    set(gca,'FontWeight','normal');
    xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
    set(xlhand,'fontsize',labelFontSize);
    set(ylhand,'fontsize',labelFontSize)
    set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
```

```
   set(xlhand,'FontWeight','bold');
   set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
   set(ylhand,'FontWeight','bold');
   set(gcf, 'paperunits', 'centimeters')

   % Save Figure
   % saveas(gcf,strcat(FigDir, Name), 'fig');
   set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 18])
   print(printAs,'-r450', [ResultsFolder,Name])

%% ------------------------- POLAR ANGLE -------------------------------- %
   Name=['HistPlotPolar_Pol_dump',num2str(dump.timestep)];
   % Create Figure
   fig2=figure('Name',Name,'Position',...
       [1 scrsz(4)/2.2 scrsz(3)/2.5 scrsz(4)/2],'Visible',FigVisible);
   h1=polar(dump.Section(1).FibOrient.phi(:,2),...
         dump.Section(1).FibOrient.zDist(:,1),...
         '^');hold on;
         set(h1,'Color',[1 0.5 0])
   h2=polar(dump.Section(2).FibOrient.phi(:,2),...
         dump.Section(2).FibOrient.zDist(:,1),...
         'o');hold on;
         set(h2,'Color',[0 0 1])
   h3=polar(dump.Section(3).FibOrient.phi(:,2),...
         dump.Section(3).FibOrient.zDist(:,1),...
         's');hold on;
         set(h3,'Color',[1 0 0])
   h4=polar(dump.Section(4).FibOrient.phi(:,2),...
         dump.Section(4).FibOrient.zDist(:,1),...
          'd');hold off;
          set(h4,'Color',[0 0 0])

legendString={[preAfterSymbol,'1^{st} section',preAfterSymbol],...
     [preAfterSymbol,'2^{nd} section',preAfterSymbol],...
     [preAfterSymbol,'3^{rd} section',preAfterSymbol],...
     [preAfterSymbol,'4^{th} section',preAfterSymbol]};

legend(legendString,'Location','NorthEastOutside','Orientation','vertical')

if(  strcmp(interpreterName, 'latex') )
 legend(legendString,...
      'Location','NorthEastOutside','Orientation','vertical',...
      'interpreter',interpreterName,'FontSize',labelFontSize)
end
   title([preAfterSymbol,'\phi',preAfterSymbol,' - polar angle'],...
       'interpreter',interpreterName,'FontSize',fontSizeTitle )
   grid off;
   box on;

   xlabel([preAfterSymbol,'r^{+}',preAfterSymbol,...
          ' : distance from the centre ',...
          'with ',preAfterSymbol,'r^{+}',preAfterSymbol,...
           ' = 1 at the wall'],...
   'interpreter',interpreterName,'fontsize',labelFontSize)

   set(0,'defaultaxesfontsize',14);
   set(0,'defaultaxesfontsize',stdTextFontSize);
   set(0,'defaulttextfontsize',stdTextFontSize);
   set(gca,'FontSize',stdTextFontSize);
   set(gca,'FontWeight','normal');
   xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
   set(xlhand,'fontsize',labelFontSize);
   set(ylhand,'fontsize',labelFontSize)
   set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
   set(xlhand,'FontWeight','bold');
   set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
   set(ylhand,'FontWeight','bold');
   set(gcf, 'paperunits', 'centimeters')

%% --------------------- Save Figure ------------------------------------ %
   % saveas(gcf,strcat(FigDir, Name), 'fig');
   set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 18])
   print(printAs,'-r450', [ResultsFolder,Name])

PlotStatus=1;
end
```

```
function [HistAzimuthalAngle,HistPolarAngle]=...
    func_PlotHist(dump,theta,phi,ResultsFolder);
%func_PlotHist(theta,phi,ResultsFolder)

global FigVisible interpreterName printAs

%% Get Screen Size
%scrsz = get(groot,'ScreenSize');% MATLAB >R2014b
scrsz = get(0,'ScreenSize'); % MATLAB <R2014b

%% --------------------------- Formating -------------------------------- %
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

%% ------------------------ AZIMUTHAL ANGLE ----------------------------- %
y=theta(:,2)/pi*180;
x=-180:15:180;
barWidth=0.8;

% Create Figure
Name=['HistPlot_Azi_dump',num2str(dump.timestep)];
FigHist=figure('Name',Name,'Position',...
    [10 10 600 600],'Visible',FigVisible);
HistAzimuthalAngle=hist(y,x);
HistAzimuthalAngle_Norm = HistAzimuthalAngle ./ sum(HistAzimuthalAngle);
bar(x,HistAzimuthalAngle_Norm,barWidth);
colormap(gray);

xlabel([preAfterSymbol,'\theta',preAfterSymbol,' - azimuthal angle'],...
       'interpreter',interpreterName)
ylabel([preAfterSymbol,'\Delta Q_0',preAfterSymbol],...
       'interpreter',interpreterName)

grid off;
box on;
legend boxoff;
xlim([-180-2 180+2]);
% ylim([0 1]);
set(gca,'XTick',[-180:45:180])
% set(gca,'YTick',[0:0.2:1])
set(gca,'XMinorTick','on','YMinorTick','on')

set(0,'defaultaxesfontsize',stdTextFontSize);
set(0,'defaulttextfontsize',stdTextFontSize);
set(gca,'FontSize',stdTextFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',labelFontSize); set(ylhand,'fontsize',labelFontSize)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

% Save Figure
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 18])
print(printAs,'-r450', [ResultsFolder,Name])

%% -------------------------- POLAR  ANGLE ------------------------------ %
y=phi(:,2)/pi*180;
x=0:15:180;

% Create Figure
Name=['HistPlot_Pol_dump',num2str(dump.timestep)];
FigHist=figure('Name',Name,'Position',...
    [10 10 600 600],'Visible',FigVisible);
HistPolarAngle=hist(y,x);
HistPolarAngle_Norm = HistPolarAngle ./ sum(HistPolarAngle);
bar(x,HistPolarAngle_Norm,barWidth/2);
colormap(gray);

xlabel([preAfterSymbol,'\phi',preAfterSymbol,' - polar angle'],...
       'interpreter',interpreterName,'FontSize',labelFontSize)
ylabel([preAfterSymbol,'\Delta Q_0',preAfterSymbol],...
       'interpreter',interpreterName,'FontSize',labelFontSize)

grid off;
box on;
legend boxoff;
xlim([-1 180+1]);
% ylim([0 1]);
set(gca,'XTick',[0:45:180])
% set(gca,'YTick',[0:0.2:1])
set(gca,'XMinorTick','on','YMinorTick','on')

set(0,'defaultaxesfontsize',stdTextFontSize);
set(0,'defaulttextfontsize',stdTextFontSize);
set(gca,'FontSize',stdTextFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',labelFontSize); set(ylhand,'fontsize',labelFontSize)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

%% --------------------- Save Figure ----------------------------------- %
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 20 18])
print(printAs,'-r450', [ResultsFolder,Name])
end
```

```matlab
function [xAzi, barDataYAzi, barDataYNormAzi, xPol, barDataYPol,...
    barDataYNormPol]=func_PlotHist_FibClass(dump,theta,phi,ResultsFolder);

% function [xAzi, barDataYNormAzi, barDataYAzi, xPol, barDataYNormPol,...
%     barDataYNormPol]=func_HistAngleFibLength(theta,phi,ResultsFolder);

%% Angle Distrubition with Fibre Distribution
global FigVisible interpreterName printAs dMinor

%% Get Screen Size
%scrsz = get(groot,'ScreenSize');% MATLAB >R2014b
scrsz = get(0,'ScreenSize'); % MATLAB <R2014b

%% ---------------------------- Formating ------------------------------- %
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

%% --------------------- Pre-Calculations ------------------------------- %
theta=sortrows(theta,1);
thetaIn=theta(1,1);

phi=sortrows(phi,1);
phiIn=phi(1,1);

% Dividing fibres in Groups acc. to their length!
% here tri-dispers system, in case of polydispers system, classes need to
% be defined!
LenGroup=1;
Counti=0;
for thetai=1:length(theta)   ;
    if theta(thetai) == thetaIn;
        Counti=Counti+1;
        thetaLenSort(LenGroup).Data(Counti,:)=theta(thetai,:);
        phiLenSort(LenGroup).Data(Counti,:)=phi(thetai,:);
    else
        Counti=1;
        LenGroup=LenGroup+1;
        thetaIn=theta(thetai);
        thetaLenSort(LenGroup).Data(Counti,:)=theta(thetai,:);
        phiLenSort(LenGroup).Data(Counti,:)=phi(thetai,:);
    end
end
nLenGroup=LenGroup;

barDataYAzi=[];
xAzi=-180:15:180;

barDataYPol=[];
xPol=0:15:180;

for LenGroup=1:nLenGroup
thetaLenSort(LenGroup).HistData=...
    histcounts(thetaLenSort(LenGroup).Data(:,2)/pi*180,xAzi);
phiLenSort(LenGroup).HistData=...
    histcounts(phiLenSort(LenGroup).Data(:,2)/pi*180,xPol);

barDataYAzi=cat(1,barDataYAzi, thetaLenSort(LenGroup).HistData);
barLedgendAzi(LenGroup)=...
    {[preAfterSymbol,'AR=',...
      num2str(thetaLenSort(LenGroup).Data(1)*2/dMinor),preAfterSymbol]};

barDataYPol=cat(1,barDataYPol, phiLenSort(LenGroup).HistData);
barLedgendPol(LenGroup)=...
    {[preAfterSymbol,'AR=',...
    num2str(phiLenSort(LenGroup).Data(1)*2/dMinor),preAfterSymbol]};

end

% Normalized Data (Sum =!= 1)
barDataYNormAzi=barDataYAzi./sum(sum(barDataYAzi));
barDataYNormPol=barDataYPol./sum(sum(barDataYPol));

%% --------------------- AZIMUTHAL ANGEL ------------------------------- %
barWidth=0.8;
% Create Figure
Name=['HistPlot_Azi_FibClasses_dump',num2str(dump.timestep)];
FigHistFibLen1=figure('Name',Name,'Position',...
    [1 scrsz(4)/2.2 scrsz(3)/2.5 scrsz(4)/2],'Visible',FigVisible);

for xi=1:1:length(xAzi)-1
xAziBar(xi)=xAzi(xi)+(xAzi(xi+1)-xAzi(xi))/2;
end

% Every 3th entry is a labeled xtick
XTickLabelAzi=ones(1,length(xAzi))*NaN;
XTickLabelAzi(1:3:end)=xAzi(1:3:end);
XTickLabelAzi=num2cell(XTickLabelAzi);
% Replace NaN with blank space
XTickLabelAzi(cellfun(@(x) any(isnan(x)),XTickLabelAzi)) = {''};

bar(xAziBar,barDataYNormAzi.',barWidth,'stacked');

 legend(barLedgendAzi,'Location','northeast','Orientation','vertical')

 if(  strcmp(interpreterName, 'latex') )
   legend(barLedgendAzi,'Location','northeast',...
   'interpreter',interpreterName,...
    'FontSize',fontSizeLegend,'Orientation','vertical')
 end


%title('atimuthal angle Hist Fibre Length')
grid off;
box on;
legend boxoff;
legend off;

xlabel([preAfterSymbol,'\theta',preAfterSymbol,' - azimuthal angle'],...
        'interpreter',interpreterName,'FontSize',labelFontSize);
ylabel([preAfterSymbol,'\Delta Q_0',preAfterSymbol],...
        'interpreter',interpreterName,'FontSize',labelFontSize);
colormap (gray); %lines(5); %(gray);

% yMaxVal=round(max(max(barDataYNormAzi))*10)/10;

xlim([-180-10 180+10])
% ylim([0 yMaxVal*1.5])
set(gca,'XTick',[-180:15:180])
set(gca,'XTickLabel',XTickLabelAzi)
% set(gca,'XMinorTick','on','YMinorTick','on')
set(0,'defaultaxesfontsize',stdTextFontSize);
set(0,'defaulttextfontsize',stdTextFontSize);
set(gca,'FontSize',stdTextFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',labelFontSize); set(ylhand,'fontsize',labelFontSize)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

% Save Figure
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 18])
print(printAs,'-r450', [ResultsFolder,Name])

%% --------------------- POLAR ANGLE ----------------------------------- %
% Create Figure
```

```matlab
function [xAzi, barDataYAzi, barDataYNormAzi, xPol, barDataYPol,...
barDataYNormPol]=func_PlotHistPolar_FibClass(dump,theta,phi,ResultsFolder);

% function [xAzi, barDataYNormAzi, barDataYAzi, xPol, barDataYNormPol,...
%     barDataYNormPol]=func_HistAngleFibLength(theta,phi,ResultsFolder);

%% Angle Distrubition with Fibre Distribution
global FigVisible interpreterName printAs
%% Get Screen Size
%scrsz = get(groot,'ScreenSize');% MATLAB >R2014b
scrsz = get(0,'ScreenSize'); % MATLAB <R2014b

%% --------------------------- Formating ------------------------------ %
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

%% ----------------------- Pre-Calculations --------------------------- %
theta=sortrows(theta,1);
thetaIn=theta(1,1);

phi=sortrows(phi,1);
phiIn=phi(1,1);

% Dividing fibres in Groups acc. to their length!
% here tri-dispers system, in case of polydispers system, classes need to
% be defined!
LenGroup=1;
Counti=0;
for thetai=1:length(theta);
    if theta(thetai) == thetaIn;
        Counti=Counti+1;
        thetaLenSort(LenGroup).Data(Counti,:)=theta(thetai,:);
        phiLenSort(LenGroup).Data(Counti,:)=phi(thetai,:);
    else
        Counti=1;
        LenGroup=LenGroup+1;
        thetaIn=theta(thetai);
        thetaLenSort(LenGroup).Data(Counti,:)=theta(thetai,:);
        phiLenSort(LenGroup).Data(Counti,:)=phi(thetai,:);
    end
end
nLenGroup=LenGroup;

barDataYAzi=[];
xAzi=-180:15:180;
xAziPlot=xAzi/180*pi;

barDataYPol=[];
xPol=0:15:180;
xPolPlot=xPol/180*pi;

for LenGroup=1:nLenGroup
thetaLenSort(LenGroup).HistData=...
    hist(thetaLenSort(LenGroup).Data(:,2)/pi*180,xAzi);
phiLenSort(LenGroup).HistData=...
    hist(phiLenSort(LenGroup).Data(:,2)/pi*180,xPol);

barDataYAzi=cat(1,barDataYAzi, thetaLenSort(LenGroup).HistData);
barLedgendAzi(LenGroup)={[preAfterSymbol,'l_{Fib}',preAfterSymbol,...
                    ' = ',num2str(thetaLenSort(LenGroup).Data(1)*2)]};

barDataYPol=cat(1,barDataYPol, phiLenSort(LenGroup).HistData);
barLedgendPol(LenGroup)={[preAfterSymbol,'l_{Fib}',preAfterSymbol,...
                    ' = ',num2str(phiLenSort(LenGroup).Data(1)*2)]};

% Normalized data within one Length Class
thetaLenSort(LenGroup).HistDataNorm = thetaLenSort(LenGroup).HistData...
    ./ sum(thetaLenSort(LenGroup).HistData);
```

```matlab
Name=['HistPlot_Pol_FibClasses_dump',num2str(dump.timestep)];
FigHistFibLen2=figure('Name',Name,'Position',...
    [1 scrsz(4)/2.2 scrsz(3)/2.5 scrsz(4)/2],'Visible',FigVisible);

for xi=1:1:length(xPol)-1
xPolBar(xi)=xPol(xi)+(xPol(xi+1)-xPol(xi))/2;
end

% Every 3th entry is a labeled xtick
XTickLabelPol=ones(1,length(xPol))*NaN;
XTickLabelPol(1:3:end)=xPol(1:3:end);
XTickLabelPol=num2cell(XTickLabelPol);
% Replace NaN with blank space
XTickLabelPol(cellfun(@(x) any(isnan(x)),XTickLabelPol)) = {''};

bar(xPolBar,barDataYNormPol.',barWidth/2,'stacked')

 legend(barLedgendPol,'Location','northeast','Orientation','vertical')

 if(  strcmp(interpreterName, 'latex') )
   legend(barLedgendPol,'Location','northeast',...
   'interpreter',interpreterName,...
   'FontSize',fontSizeLegend,'Orientation','vertical')
 end

%title('polar angle Hist Fibre Length')
grid off;
box on;
legend boxoff

xlabel([preAfterSymbol,'\phi',preAfterSymbol,' - polar angle'],...
    'interpreter',interpreterName,'FontSize',labelFontSize)
ylabel([preAfterSymbol,'\Delta Q_0',preAfterSymbol],...
    'interpreter',interpreterName,'FontSize',labelFontSize)
colormap (gray); %lines(5); %(gray);

xlim([0-5 180+5])
set(gca,'XTick',[0:15:180])
set(gca,'XTickLabel',XTickLabelPol)
% set(gca,'XMinorTick','on','YMinorTick','on')
set(0,'defaultaxesfontsize',stdTextFontSize);
set(0,'defaulttextfontsize',stdTextFontSize);
set(gca,'FontSize',stdTextFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',labelFontSize); set(ylhand,'fontsize',labelFontSize)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

%% --------------------- Save Figure ------------------------------------ %
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 18])
print(printAs,'-r450', [ResultsFolder,Name])

end
```

```
phiLenSort(LenGroup).HistDataNorm = phiLenSort(LenGroup).HistData...
   ./ sum(phiLenSort(LenGroup).HistData);
end

% Normalized Data over all Length Classes (Sum =!= 1)
barDataYNormAzi=barDataYAzi./sum(sum(barDataYAzi)); % dQ0
barDataYNormPol=barDataYPol./sum(sum(barDataYPol)); % dQ0

%% ------------------------ AZIMUTHAL ANGEL ----------------------------- %
% Create Figure
Name=['HistPlotPolar_Azi_FibClasses_dump',num2str(dump.timestep)];
FigHistFibLen1=figure('Name',Name,'Position',...
   [1 scrsz(4)/2.2 scrsz(3)/2.5 scrsz(4)/2],'Visible',FigVisible);

 cc = jet(nLenGroup);
 if(  strcmp(interpreterName, 'latex') )
  cc=lines(nLenGroup);
 end

for LenGroup=1:1:nLenGroup
x=xAziPlot(:).';
y=thetaLenSort(LenGroup).HistDataNorm; %barDataYNormAzi(LenGroup,:)

% First Value double at end to close Curve, !spline-error -> no value twice
% -> change first value by *(1.0+1e-6)
if x(1) == 0.0
   x=[x,1e-8];
else
   x=[x,(x(1)*(1.0+1e-8))];
end

if y(1) == 0.0
    y=[y,1e-8];
else
    y=[y,(y(1)*(1.0+1e-8))];
end

yy=spline(x,y,x);
h1=polar(x,yy); hold on;
set(h1,'Color',cc(LenGroup,:))
end

legend(barLedgendAzi,'Location','NorthEastOutside',...
    'Orientation','vertical')

 if(  strcmp(interpreterName, 'latex') )
legend(barLedgendAzi,'Location','NorthEastOutside',...
    'interpreter','latex',...
    'FontSize',fontSizeLegend,'Orientation','vertical')
 end


title([preAfterSymbol,'\theta',preAfterSymbol,...
       ' - azimuthal angle'],...
      'interpreter',interpreterName,'FontSize',fontSizeTitle);
xlabel(['radius ... ',preAfterSymbol,'\Delta Q_0',preAfterSymbol],...
       'interpreter',interpreterName,'FontSize',labelFontSize);

grid off;
box on;

set(0,'defaultaxesfontsize',stdTextFontSize);
set(0,'defaulttextfontsize',stdTextFontSize);
set(gca,'FontSize',stdTextFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',labelFontSize); set(ylhand,'fontsize',labelFontSize)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters');

%% ------------------------ Save Figure --------------------------------- %
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 18])
print(printAs,'-r450', [ResultsFolder,Name])

%% ------------------------ POLAR ANGLE --------------------------------- %
% Create Figure
Name=['HistPlotPolar_Pol_FibClasses_dump',num2str(dump.timestep)];
FigHistFibLen2=figure('Name',Name,'Position',...
   [1 scrsz(4)/2.2 scrsz(3)/2.5 scrsz(4)/2],'Visible',FigVisible);

 cc = jet(nLenGroup);
 if(  strcmp(interpreterName, 'latex') )
  cc=lines(nLenGroup);
 end

for LenGroup=1:1:nLenGroup
x=xPolPlot(:).';
y=phiLenSort(LenGroup).HistDataNorm;  %barDataYNormPol(LenGroup,:)

% First Value double at end to close Curve, !spline-error -> no value twice
% -> change first value by *(1.0+1e-6), id first value == 0 -> + 1e-8
if x(1) == 0.0
   x=[x,1e-8];
else
   x=[x,(x(1)*(1.0+1e-8))];
end

if y(1) == 0.0
   y=[y,1e-8];
else
   y=[y,(y(1)*(1.0+1e-8))];
end

yy=spline(x,y,x);
h2=polar(x,yy); hold on;
set(h2,'Color',cc(LenGroup,:))
end

legend(barLedgendPol,'Location','NorthEastOutside','Orientation','vertical')

 if(  strcmp(interpreterName, 'latex') )
legend(barLedgendPol,'Location','NorthEastOutside','interpreter','latex',...
   'FontSize',fontSizeLegend,'Orientation','vertical')
 end

title([preAfterSymbol,'\phi',preAfterSymbol,...
       ' - polar angle'],...
       'interpreter',interpreterName,'FontSize',fontSizeTitle);
xlabel(['radius ...',preAfterSymbol,'\Delta Q_0',preAfterSymbol],...
        'interpreter',interpreterName,'FontSize',labelFontSize);

grid off;
box on;

set(0,'defaultaxesfontsize',stdTextFontSize);
set(0,'defaulttextfontsize',stdTextFontSize);
set(gca,'FontSize',stdTextFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',labelFontSize); set(ylhand,'fontsize',labelFontSize)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters');

%% ------------------------ Save Figure --------------------------------- %
```

```
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 18])
print(printAs,'-r450', [ResultsFolder,Name])

end
```

```
function [Section]=func_Section(dump)
global resDirMain resDirSub FigVisible printAs interpreterName
% Devide Fibres acc. to their position in 4 sectors
% Basis: (x) y z from dump file (global coordinate system from LIGGGHTS)
%
%         +-------------+
%         |\     3     /|
%         | \         / |
%         |   \     /   |
%         | 2     o     4 |
%         |   /     \   |
%         | /         \ |
%         |/     1     /|
%         +-------------+
%

%% ---------------------------- Formating ----------------------------- %
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

% Boundaries:
yMin=dump.y_bound(1); yMax=dump.y_bound(2);
zMin=dump.z_bound(1); zMax=dump.z_bound(2);

yP = dump.atom_data(:,4);
zP = dump.atom_data(:,5);

Section(1).Count=0; Section(2).Count=0;
Section(3).Count=0; Section(4).Count=0;

% Devide in Sections
for i=1:1:length(yP);

    % Sector 1 with (0,0) is in Section 1 due to order
      if or(yMin <   yP(i) & yP(i) <= 0 & ...
            zMin <=  zP(i) & zP(i) < -abs(yP(i)),...          % or
            0    <=  yP(i) & yP(i) <= yMax & ...
            zMin <=  zP(i) & zP(i) <= -abs(yP(i)))
         %disp('sec 1');
         Section(1).Count=Section(1).Count+1;
         Section(1).atom_data_G(Section(1).Count,:)=dump.atom_data(i,:);

    % Sector 2
      elseif or( 0           <=  yP(i) & yP(i) <= yMax & ...
                 -abs(yP(i)) <   zP(i) & zP(i) <= 0 ,...      % or
                 0           <=  yP(i) & yP(i) <= yMax & ...
                 0           <=  zP(i) & zP(i) <= abs(yP(i)))
             %disp('sec 2');
             Section(2).Count=Section(2).Count+1;
             Section(2).atom_data_G(Section(2).Count,:)=dump.atom_data(i,:);

     % Sector 3
      elseif or(0          <=  yP(i) & yP(i) <= yMax & ...
                abs(yP(i)) <   zP(i) & zP(i) <= zMax,...      % or
                yMin       <=  yP(i) & yP(i) <= 0 & ...
                abs(yP(i)) <=  zP(i) & zP(i) <= zMax)
             %disp('sec 3');
             Section(3).Count=Section(3).Count+1;
             Section(3).atom_data_G(Section(3).Count,:)=dump.atom_data(i,:);

      % Sector 4
      elseif or(yMin  <=  yP(i) & yP(i) <= 0 & ...
                0     <=  zP(i) & zP(i) < abs(yP(i)),...        % or
                yMin  <=  yP(i) & yP(i) <= 0 & ...
                -abs(yP(i))     <=  zP(i) & zP(i) <= 0)
             %disp('sec 4');
             Section(4).Count=Section(4).Count+1;
             Section(4).atom_data_G(Section(4).Count,:)=dump.atom_data(i,:);
    end
end

% Plot: Fibres in Cross Section Sections
scrsz = get(0,'ScreenSize');
fig1=figure('Position',[10 10 600 600],...
    'visible',FigVisible);

plot(Section(1).atom_data_G(:,4),Section(1).atom_data_G(:,5),...
     '^','Color',[1 0.5 0]); hold on;    % orange
plot(Section(2).atom_data_G(:,4),Section(2).atom_data_G(:,5),...
     'o','Color',[0 0 1]); hold on; % blue
plot(Section(3).atom_data_G(:,4),Section(3).atom_data_G(:,5),...
     's','Color',[1 0 0]); hold on;     % red
plot(Section(4).atom_data_G(:,4),Section(4).atom_data_G(:,5),...
     'd','Color',[0 0 0]); hold on;     % black
plot([yMin,yMax],[zMin,zMax],'k-'); hold on;
plot([yMin,yMax],[zMax,zMin],'k-')
 set(gca,'XDir','reverse');

 legendString={[preAfterSymbol,'1^{st} section',preAfterSymbol],...
          [preAfterSymbol,'2^{nd} section',preAfterSymbol],...
          [preAfterSymbol,'3^{rd} section',preAfterSymbol],...
          [preAfterSymbol,'4^{th} section',preAfterSymbol]};

legend(legendString,'Location','northeast','Orientation','vertical')

 if(  strcmp(interpreterName, 'latex') )
  legend(legendString,...
      'Location','northeast','Orientation','vertical',...
      'interpreter',interpreterName,'FontSize',labelFontSize)
 end

 xlabel([preAfterSymbol,'y-direction',preAfterSymbol],...
     'interpreter',interpreterName)
 ylabel([preAfterSymbol,'z-direction',preAfterSymbol],...
     'interpreter',interpreterName)
 title('Fibres in different cross section sections',...
     'interpreter',interpreterName)

 axis([yMin yMax zMin zMax])
 axis square

box on;
set(0,'defaultaxesfontsize',stdTextFontSize);
set(0,'defaulttextfontsize',stdTextFontSize);
set(gca,'FontSize',stdTextFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',labelFontSize); set(ylhand,'fontsize',labelFontSize)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')


%% ------------------------ Save Figure --------------------------------- %
Name=['CrossSection_dump',num2str(dump.timestep),'_ligggths'];
 % saveas(gcf,strcat(FigDir, Name), 'fig');
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 18])
 print(printAs,'-r450', [fullfile(resDirMain,resDirSub),Name])

end
```

```
% If gnuplot/ octave is used for plots
global interpreterName

fontSizeTitle=16;
fontSizeLegend=14;
fontSizeAxis=20;
markerSize = 9;

lineWidth=2;
stdTextFontSize  = 16;
labelFontSize    = 18;

if(  strcmp(interpreterName, 'tex') )
   graphics_toolkit gnuplot
   set (0, "defaultaxesfontname", "TimesItalic")
   set (0, "defaultaxesfontsize", stdTextFontSize)
   set (0, "defaulttextfontname", "TimesItalic")
   set (0, "defaultTextFontSize", stdTextFontSize)
   preAfterSymbol = '';
end
```

```
% If MATLAb is used for plots
fontSizeTitle=16;
fontSizeLegend=18;
fontSizeAxis=20;
markerSize = 9;

lineWidth=2;
stdTextFontSize  = 18;
labelFontSize    = 18;

preAfterSymbol = '';
if(  strcmp(interpreterName, 'latex') )
    preAfterSymbol = '$';
end

```

```
function [theta,phi]=func_ConvCart2Polar(xVec)
% [theta,phi]=func_ConvCart2Polar(xVec)
% (c) Lisa KÃ¶nig
% Function to convert cartasian coordinates to polar coordinates

x = xVec(1);
y = xVec(2);
z = xVec(3);

% Calc theta ... azimuthal angle
% differ cases
if      x > 0
        theta=atan(y/x);
elseif  and (x < 0 , y >= 0)
        theta=atan(y/x)+pi;
elseif  and (x < 0 , y < 0)
        theta=atan(y/x)-pi;
elseif  and (x == 0 , y > 0)
        theta=pi/2;
elseif  and (x == 0 , y < 0)
        theta=-pi/2;
elseif  and (x == 0 , y == 0) % exact z-direction -> mathematic.undfefined
        theta=pi/2;
end

% Calc phi ... polar angle
phi=acos(z);

end
```

```matlab
function [D]=func_GeneralRotationMatrix(rotVec,alpha)
%[D]=func_GeneralRotationMatrix(rotVec,alpha)
% (c) Lisa Maria Koenig, April 2015

% Calculate the general rotation Matrix (germ. Allgemeine Drehmatrix, in
% Matrixform)
% D       ... general rotation matrix
% rotVec ... unit rotation vector
% alpha  ... rotation angle in rad

% Note
% To rotate a vector (aVec) in a coordinate system:
% aVec' = D * aVec
%
% To change basis and get vector (aVec) in new basis coordinates
% (rotate coordinate system, vector doesn't change/rotate!)
% aVec_newBasis = transpose(D) * aVec_oldBasis
%
% This function is an implementation of the general rotation matrix from
% the source: Book: Otto - Rechenmethoden fuer Studierende der Pysik im
% ersten Jahr p.79
%

% General rotation matrix in matrix format
D = [ cos(alpha)      + (1-cos(alpha)) * rotVec(1)^2 ...
      (1-cos(alpha)) * rotVec(1) * rotVec(2) + sin(alpha) * rotVec(3)...
      (1-cos(alpha)) * rotVec(1) * rotVec(3) - sin(alpha) * rotVec(2);...

      (1-cos(alpha)) * rotVec(1) * rotVec(2) - sin(alpha) * rotVec(3)...
       cos(alpha)      + (1-cos(alpha)) * rotVec(2)^2 ...
      (1-cos(alpha)) * rotVec(2) * rotVec(3) + sin(alpha) * rotVec(1);...

      (1-cos(alpha)) * rotVec(1) * rotVec(3) + sin(alpha) * rotVec(2)...
      (1-cos(alpha)) * rotVec(2) * rotVec(3) - sin(alpha) * rotVec(1)...
       cos(alpha)      + (1-cos(alpha)) * rotVec(3)^2];

end
```

```matlab
% Program to write dump-data in format for read_data in LIGGHTS(R)
% Provide required intpu (see below)
% function func_writeParticleData is called.
% (c) Lisa Koenig, TU Graz,
%
% Required dump:
% 1            ... id
% 2            ... type
% 3 4 5        ... x y z
% 6 7 8        ... vx vy vz
% 9 10 11      ... fx fy fz
% 12 13 14     ...omegax omegay omegaz
% 15           ... radius
% 16 17 18     ... f_ex[1] f_ex[2] f_ex[3]
% 19           ... f_shape[1]
% 20 21 22 23 ...c_cQuatw c_cQuati c_cQuatj c_cQuatk
clear all; close all; clc
global homeDir dumpDir resDir nameOut caseIdToRun

%% -------------------- User Input --------------------------------------%
% Directories
homeDir=pwd;                              % current directory
resDir='../../';                          % directory for result file
nameOut='in.Particle_Flock4CFDEM';       % name for output file .data

% Input
input.setRegion  = 1;       % 0 ... region values from dump
                            % 1 ... own region values as below
input.regionMinX = -2.50;  % simulation domain minimum in x-direction
input.regionMaxX =  6.50;  % simulation domain maximum in x-direction
input.regionMinY = -0.50;  % simulation domain minimum in y-direction
input.regionMaxY =  0.50;  % simulation domain maximum in y-direction
input.regionMinZ = -4.50;  % simulation domain minimum in z-direction
input.regionMaxZ =  0.50;  % simulation domain maximum in z-direction

input.density     = 1.270;  % all fibres input.density
input.dMinor      = 0.02;  % all fibres input.dMinor
input.dMinorRealFact=1.1;  % dMinor in read_data file differs from dMajor used for deform
                             % read_data-dMajor=dMinor/dMinorRealFactor

input.xOffSet         = -1.5; % offset of origin in x direction
input.TypeStart       =  2;   % start with this atom type number
input.setInitFibVelo =  1;   % 0 ... no, 1 ... yes
input.UmaxSqu         =  0.5; % max. velocity in laminar profil
input.halfLength      =  0.5; % Hald length of the square cross section area
input.check4Last      =  1;   % 1...last dump in directory; 0...only dump in dir

%% -------------------- Read temporary file for setting ----------------- %
% Check if octave of matlab is used
tempDataFile = fopen('tempData.txt');
C = textscan(tempDataFile,'%f %f',1);
fclose(tempDataFile);
interPreterName=C{1,1};
caseIdToRun=C{1,2};

if interPreterName == 1 % MATLAB
    interpreterName='latex';
end

if interPreterName == 2 %OCTAVE
    interpreterName='tex';
end

% directory of the dump-file
dumpDir=['../../post_Case_',num2str(caseIdToRun)];                % directory of the dump-file


%% -------------------- Run Function ------------------------------------%
funcStat=func_writeParticleData(input);

```

```matlab
function [vel] = velSCS(s, UmaxSqu, r, theta)
% velDistributionPolygonialDuct - calculates the velocity distribution in a
% polygonial duct. Velocity scaled to give a mean velocity Umean
% Approach based on Tamayol and Bahrami, J Fluids Engineering 132, 111201
% [vel] = velDistributionPolygonialDuct(m, s, Umax, r, theta)
% INPUT
%    m    ...number of sides (4...square)
%    s    ...half side length
%    Umax ...maximal velocity
%    r    ...radial distance
%    theta...angular position of the point
% OUTPUT
%    vel  ...the velocity at the point (r, theta)

m=4;

A1=0.247+0.767/m^2;
C(1)=(6.01-3.12*m-0.965*m^2)^-1;
if(m==4)
    C(2)=-1.096e-3;
end

eta = r .* tan(pi/m) ./ s;
uStar = zeros(size(eta,1),size(eta,2));
for i=1:size(eta,1)
    for j=1:size(eta,2)
        currEta=eta(i,j);
        currTheta=theta(i,j);
        uStar(i,j) = 1-currEta^2/4/A1;
        for it=1:length(C)
            uStar(i,j)=uStar(i,j)+C(it)/A1*currEta^(m*it)*cos(m*currTheta);
        end
    end
end

vel = UmaxSqu*uStar;
end
```

```matlab
%% -------------------- End of Program -----------------------------------%
disp('End of Program. Have fun ...')

```

```matlab
function funcStat=func_writeParticleData(input)
% function to write read_data input file
global homeDir dumpDir resDir nameOut caseIdToRun

%% ---------------- Find Last Dump File in dump  Directory ---------------%
if input.check4Last==1
  % Check number of dump-files in folder
    cd(dumpDir);
     files=dir('dump*.liggghts'); % list all dump-files
     cd(homeDir);
     nDmpFiles=length(files);

    % List all Numbers of the dump-files and find first and last
    for i=1:1:nDmpFiles
      files(i).numbers=sscanf(files(i).name,'dump%i.liggghts');
    end

    LstDump=max([files.numbers]);  % last Dump File (e.g. end of sim)
    Name=['dump',num2str(LstDump,'%i'),'.liggghts'];

else
    % get name from dump-file from directory
     file=dir(fullfile(dumpDir,'dump*'));
     Name=file.name
end
disp(['Dump to read: dump',num2str(LstDump,'%i'),'.liggghts'])


%% ---------------- Read Dump-File Data ---------------%
data = readdump_all(fullfile(dumpDir,Name));

% Set Simulation Region
    if input.setRegion == 0 % as in dump File
       input.regionMinX = data.x_bound(1,1);  % simulation domain minimum in x-direction
       input.regionMaxX = data.x_bound(1,2);  % simulation domain maximum in x-direction
       input.regionMinY = data.y_bound(1,1);  % simulation domain minimum in y-direction
       input.regionMaxY = data.y_bound(1,2);  % simulation domain maximum in y-direction
       input.regionMinZ = data.z_bound(1,1);  % simulation domain minimum in z-direction
       input.regionMaxZ = data.z_bound(1,2);  % simulation domain maximum in z-direction
    end
    % else as in input

%% Needed Data:
nAtoms = size(data.atom_data,1);                % no. atoms in system
nEllipsoids = nAtoms;                           % no. ellipsoids in system
nTypes = max(data.atom_data(:,2))+input.TypeStart-1; % no. atom types

ID   = data.atom_data(:,1);                     % ID
Type = data.atom_data(:,2)+input.TypeStart-1;         % Type
xVal = data.atom_data(:,3)+input.xOffSet;             % x position
yVal = data.atom_data(:,4);                          % yosition
zVal = data.atom_data(:,5);                          % z position
diam = (input.dMinor.*input.dMinor.*...
        2.*data.atom_data(:,19)).^0.333333333333333333; % Vol. equi. diameter
input.density = ones(nAtoms,1)*input.density;         % fibre input.density
ellipsoidflag = ones(nAtoms,1)*1;                    % flag 1=ellispoid
shapeX = 2*data.atom_data(:,19);                     % main axis
shapeY = ones(nAtoms,1)*input.dMinor;                  % side axis
shapeZ = ones(nAtoms,1)*input.dMinor;                  % side axis
quatw = data.atom_data(:,20);                         % quaternion w-value
quati = data.atom_data(:,21);                         % quaternion i-value
quatj = data.atom_data(:,22);                         % quaternion j-value
quatk = data.atom_data(:,23);                         % quaternion k-value

%% Set initial velocity
if input.setInitFibVelo == 1

    yVelo = ones(nAtoms,1)*0; % velocity in y-direction
    zVelo = ones(nAtoms,1)*0; % velocity in z-direction

    omx = ones(nAtoms,1)*0; % angular velocity
    omy = ones(nAtoms,1)*0; % angular velocity
    omz = ones(nAtoms,1)*0; % angular velocity

    lx = ones(nAtoms,1)*0; % angular momentum
    ly = ones(nAtoms,1)*0; % angular momentum
    lz = ones(nAtoms,1)*0; % angular momentum

    r = sqrt(yVal.^2 + zVal.^2);
    theta = acos(yVal./r);
    [xVelo] = velSCS(input.halfLength, input.UmaxSqu, r, theta);

    for i=1:1:nAtoms

      % Elimiate NAN velue at center of Channel
       if and(yVal(i)==0,zVal(i)==0)
          xVelo(i)=input.UmaxSqu;
       end

       % Elimiate negative Values
       if xVelo(i) < 0
           xVelo(i) =0;
       end

    end
plot3(yVal,zVal,xVelo,'.')
end





%% Write Output File
% Open File to write in
File1=fopen(fullfile(resDir,[nameOut,'_',num2str(caseIdToRun),'.data']), 'w');

% #format for "Atoms" section:        id type x y z diameter density ellipsoidflag density
% #format for "Ellipsoids" section:   id shapex shapey shapez quatw  quati  quatj  quatk
% #format for "Velocities" section:   id vx vy vz lx ly lz

% Write to File
fprintf(File1,'%s \n',...
    '# LAMMPS data file for rifig bodies hybrid granular ellipsoid');
fprintf(File1,'%s \n','');
fprintf(File1,'%i %s \n',nAtoms,      '        atoms');
fprintf(File1,'%i %s \n',nEllipsoids,'        ellipsoids');
fprintf(File1,'%i %s \n',nTypes,      '        atom types');
fprintf(File1,'%s \n','');
fprintf(File1,'%4.2f %4.2f %s \n',...
    input.regionMinX,input.regionMaxX,'xlo xhi');
fprintf(File1,'%4.2f %4.2f %s \n',...
    input.regionMinY,input.regionMaxY,'ylo yhi');
fprintf(File1,'%4.2f %4.2f %s \n',...
    input.regionMinZ,input.regionMaxZ,'zlo zhi');
fprintf(File1,'%s \n','');
fprintf(File1,'%s \n','Atoms');
fprintf(File1,'%s \n','');

for j=1:1:nAtoms
fprintf(File1,'%i %i %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f \n',...
    ID(j), Type(j), xVal(j), yVal(j), zVal(j),...
    diam(j), input.density(j), ellipsoidflag(j), input.density(j));
end

fprintf(File1,'%s \n','');
fprintf(File1,'%s \n','Ellipsoids');
fprintf(File1,'%s \n','');
for j=1:1:nAtoms
fprintf(File1,'%i %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f \n',...
    ID(j), shapeX(j), shapeY(j), shapeZ(j),...
    quatw(j),quati(j),quatj(j),quatk(j));
end

if input.setInitFibVelo == 1
    fprintf(File1,'%s \n','');
    fprintf(File1,'%s \n','Velocities');
    fprintf(File1,'%s \n','');
    for j=1:1:nAtoms
    fprintf(File1,'%i %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f \n',...
        ID(j), xVelo(j), yVelo(j), zVelo(j),omx(j),omy(j),omz(j),...
        lx(j),ly(j),lz(i));
    end
end


funcStat=1;
end

```

```
% Program to determine impact on z-postion from Fibre-Wall Interaction

% (c) Lisa Koenig
%
% Required dump:
% 1           ... id
% 2           ... type
% 3 4 5       ... x y z
% 6 7 8       ... vx vy vz
% 9 10 11     ... fx fy fz
% 12 13 14    ...omegax omegay omegaz
% 15          ... radius
% 16 17 18    ... f_ex[1] f_ex[2] f_ex[3]
% 19          ... f_shape[1]
% 20 21 22 23 ...c_cQuatw c_cQuati c_cQuatj c_cQuatk

clear all; close all; clc
global inDirDump resDir homeDir figVisible

inDirDump='../../post';
resDir='../../postProcessing';
homeDir=pwd;
figVisible='on',
halfLength=0.5;  % Half Length of the Channel height
dtDEM=1e-6;
dumpFreq=25000;
timePerDump=dtDEM*dumpFreq;
Fib.dMinor=0.02;

%% -----------------------  Environment Setting ----------------------- %
% Get Interpreter and other Programm Settings
tempDataFile=fopen('tempData.txt');
tempData=textscan(tempDataFile,'%f %f',1);
fclose(tempDataFile);
caseIdToRun=tempData{1,1};
interpreterName=tempData{1,2};
clear tempData

if interpreterName == 1 % MATLAB
    interpreterName = 'latex';
end

if interpreterName == 0 % OCTAVE
    interpreterName = 'tex';
end

% Check Folders, if they don't exist create it
if exist(resDir) == 0 % if folder does not exist create it!
    mkdir(resDir);
    disp(['Folder: ', fullfile(resDir),' created']);
end

% --------------------------------
% Check number of dump-files in folder
cd(inDirDump);
files=dir('dump*.liggghts'); % list all dump-files
cd(homeDir);
nDmpFiles=length(files);

% List all name-numbers of the dump-files and find first and last
for i=1:1:nDmpFiles
FileNames(i,1)=sscanf(files(i).name,'dump%i%.liggghts');
end

% Sort File Names
FileNames=sortrows(FileNames,1);
FrstDmp=min(FileNames);
LstDmp=max(FileNames);

% Read first and last dump-file and sort atom data ac.. to ID (first row)
```

```
dmpFrst       = ...
   readdump_all(fullfile(inDirDump,['dump',num2str(FrstDmp,'%i'),'.liggghts']));
  dmpFrst.atom_data=sortrows(dmpFrst.atom_data,1);
dmpLst= ...
   readdump_all(fullfile(inDirDump,['dump',num2str(LstDmp,'%i'),'.liggghts']));
  dmpLst.atom_data=sortrows(dmpLst.atom_data,1);

% Fibres in each Class
Fib.nClass=unique(dmpFrst.atom_data(:,2))
for j=1:1:length(Fib.nClass)
Fib.nFibInClass(j)=length(find(dmpFrst.atom_data(:,2)==Fib.nClass(j)))
end

% Initial and end z-position
for i=1:1:size(dmpFrst.atom_data,1)

   ResData(i,1)= dmpFrst.atom_data(i,1); % ID
   ResData(i,2)= dmpFrst.atom_data(i,2); % Type
   ResData(i,3)= dmpLst.atom_data(i,19)*2; % dMajor
   ResData(i,4)= dmpFrst.atom_data(i,5); % z_initial
   ResData(i,5)= dmpLst.atom_data(i,5);  % z_end
   ResData(i,6)= ...
      (ResData(i,5) - ResData(i,4)) /...
     (halfLength-abs(ResData(i,4)));
   ResData(i,7)= ...
      (halfLength-abs(ResData(i,5)))/(ResData(i,3)/2);
end


%% -------------------- Figure ----------------------------------------%
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

figName = 'FibreWallImpactOnZPosition';
scrs = get(0,'screensize');
rect = [10, 10, 700, 600]; %[left, bottom, width, height]
fig1 = figure('Name',figName,'Position',rect,'Visible',figVisible);

count=1;
for j=1:1:length(Fib.nClass)
   for i=1:1:Fib.nFibInClass(j)
   x(i,j)=ResData(count,4);
   y(i,j)=ResData(count,7);
   count=count+1;
   end
   plot(x(:,j),y(:,j),symbolArrayCS{j}); hold on;
   legendText{j}=[preAfterSymbol,...
   'AR=',num2str(ResData(i*j,3)/Fib.dMinor),preAfterSymbol];
end

% % Plot Diagonal
% plot([min(ResData(:,4)),max(ResData(:,4))],...
%     [min(ResData(:,5)),max(ResData(:,5))],'k-')

legend(legendText,'Location','Northwest');

 if(  strcmp(interpreterName, 'latex') )
   legend(legendText,'fontsize',fontSizeLabel,...
      'interpreter',interpreterName,'Location','Northwest');
 end

xlabel([preAfterSymbol,'z^{+}_{init}',preAfterSymbol],...
   'interpreter',interpreterName)
ylabel([preAfterSymbol,...
   '(s-|z^{+}_{end}|)/(d_{Major_{AR}}/2)',preAfterSymbol],...
   'interpreter',interpreterName)

   axis([-0.5 -0.3 0 10])
   axis square
   set(gca,'XTick',[-0.48 -0.45 -0.4 -0.35 -0.3])
   set(gca,'YTick',[0:2:10])
   grid off
   box on
   legend boxoff
   set(gca,'XMinorTick','on','YMinorTick','on')
   set(0,'defaultaxesfontsize',defaultaxesfontsize);
   set(0,'defaulttextfontsize',defaulttextfontsize);
   set(gca,'FontSize',fontSizeAxis);
   set(gca,'FontWeight','normal');
   xlhand = get(gca,'xlabel');
   ylhand = get(gca,'ylabel');
   zlhand = get(gca,'zlabel');
   set(xlhand,'fontsize',fontSizeLabel);
   set(ylhand,'fontsize',fontSizeLabel);
   set(zlhand,'fontsize',fontSizeLabel)
   set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
   set(xlhand,'FontWeight','bold');
   set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
   set(ylhand,'FontWeight','bold');
   set(zlhand,'FontName','Times'); set(zlhand,'FontAngle','italic');
   set(zlhand,'FontWeight','bold');

% Save Figure
Name='FibreWallImpact';
print('-dpng','-r450', fullfile(resDir,Name));

% % Read all dump-files
% for dmbNbr=1:1:nDmpFiles
%     dump(dmbNbr) = ...
%       readdump_all(...
%      fullfile(inDirDump,['dump',num2str(FileNames(dmbNbr)),'.liggghts']));
%     % Sort Data in dump-File acc to ID
%     dump(dmbNbr).atom_data = sortrows(dump(dmbNbr).atom_data)
% end
%
% % % Get z* - value for each fibre, if omegay_min has been reached for the
% % % first time (e.g. after first 180 degree spin)
% % Fib.nTotal=size(dump(1).atom_data,1)
% % for i=1:1:Fib.nTotal % == ID in Dump
% %     for dmbNbr=2:1:nDmpFiles-1
% %
% %         omegayBefore=dump(dmbNbr-1).atom_data(i,13);
% %         omegay=dump(dmbNbr).atom_data(i,13);
% %         omegayAfter=dump(dmbNbr-1).atom_data(i,13);
% %
% %         if and(omegayBefore>omegay, omegay<omegayAfter)
% %          ResData(i,1)=i;                              % Atom ID
% %          ResData(i,2)=dump(dmbNbr).atom_data(i,2);% Atom Type
% %          ResData(i,3)=dump(dmbNbr).atom_data(i,19)*2;% dMajor
% %          ResData(i,4)=dump(dmbNbr).atom_data(i,5);% end z-position
% %          ResData(i,5)=dump(1).atom_data(i,5);     % initial z-position
% %          ResData(i,6)=(ResData(i,5)-ResData(i,4));% initial minus end z-pos
% %          ResData(i,7)=...
% %             (ResData(i,5)-ResData(i,4))/halfLength;% z change fraction
% %
% %          ResData(i,8)=dmbNbr*timePerDump
% %            break
% %         end
% %
% %     end
% % end
% %
% % % Check if any ID is missing
% % tempID= [1:1:Fib.nTotal];
% % IDMissing=setdiff(ResData(i,1),tempID);
% % if not(isempty(IDMissing));
% % disp('Missing IDs!!!')
% % end
```

```
%% This preprocessing program provides Fibre Data (position and velocity)
% (e.g for usage in a CFDEM simulation)
% Lisa Koenig, August 2015, TUGraz
%
%  Schematic Diagramm:
% --------------------
%
%           Z-dir.                  Z-dir.
%             /\                      /\
%             ||                      ||
%             ||                      ||
%          +--------+   +--------------------------------+
%          |       |   |                           |
%  Y-dir.<- |       |   |                           | -> X-dir.
%          |       |   |                           |
%          +--------+   +-----------+---------+----------+
%                            |       |
%                            |       |
%
% The Fibre initial velocity can be set with the Velocity profil given by:
% Tamayol, Bahrami - Laminar Flow in Mircochannels With
% Noncircular Cross Section
% setVeloX=0 ... initial fibre velocity is zero
% setVeloX=1 ... initial fibre velocity is calculated

clear all; clc; close all; fclose all
global interpreterName resDir currDir figVisible halfLength Umax printAs

%% ------------------- Directories ----------------------------------- %
inDir = '../../input/';
fileName= 'input.txt';
resDir='../../preProcessing/';
currDir=pwd;
printAs='-dpng'; % '-dpdf' 'dpng
figVisible='on';

%% ------------------- Read temporary file for setting ---------------- %
% Check if octave of matlab is used
tempDataFile = fopen('tempData.txt');
C = textscan(tempDataFile,'%f %f',1);
fclose(tempDataFile);
InterPreterName=C{1,1};

if InterPreterName == 1 % MATLAB
    interpreterName='latex'
end

if InterPreterName == 2 %OCTAVE
    interpreterName='tex'
end

%% ------------------- Prepare environment ---------------------------- %
% Check if output directory exists
cd ..
cd ..
if exist('preProcessing') == 0 % if folder does not exist create it!
   mkdir('preProcessing');
end
cd (currDir)

%% ------------------- Read Input Data -------------------------------- %

    % Get Data from input File via dlmread()
    temp.Data1 = dlmread(strcat(inDir,fileName),'',[15 0 15 5]);
    temp.Data2 = dlmread(strcat(inDir,fileName),'',[18 0 18 10]);
    temp.Data3 = dlmread(strcat(inDir,fileName),'',21,0);

    % Box Dimensions
    Box.xMin     = temp.Data1(1);
    Box.xMax     = temp.Data1(2);
    Box.yMin     = temp.Data1(3);
    Box.yMax     = temp.Data1(4);
    Box.zMin     = temp.Data1(5);
    Box.zMax     = temp.Data1(6);

    % Fibre/Fluid Data
    Fib.rho  = temp.Data2(1);
    rhoFluid = temp.Data2(2);

    % Set velocity
    Fib.nFibZ    = temp.Data2(3);
    Fib.setVeloX = temp.Data2(4);
    Umax         = temp.Data2(5);
    halfLength   = temp.Data2(6);
    Fib.x        = temp.Data2(7);
    Fib.y        = temp.Data2(8);
    Fib.zmin     = temp.Data2(9);
    Fib.zmax     = temp.Data2(10);
    Fib.startType= temp.Data2(11);

    % Fibre Data
    Fib.Data = sortrows(temp.Data3,[1 2]); % Sort acc. to dMajor
    Fib.dMajor = Fib.Data(:,1);
    Fib.dMinor = Fib.Data(:,2);

    clear temp.*; % clear temporary data


%% ------------------- Fibre Values Calculation ----------------------- %
% Fibre Classes
Fib.nAtomType = length(Fib.dMajor);
% Diameter of Spherocylinder
Fib.dCylinder =1.24.*Fib.dMinor./sqrt(log(Fib.dMajor./Fib.dMinor));
% Aspect Ratios
Fib.AR        =Fib.dMajor./Fib.dMinor;
% Vol. single fibre in class i; fibre assumed to be ellipsoid
Vol.Fibi      = Fib.dMajor.^3 ./ Fib.AR .^2 .* pi ./ 6;
% mass of one single fibre in class i
mass.Fibi     = Fib.rho * Vol.Fibi;
% vol. equiv. diameter
Fib.dSphere   = (Fib.dMinor.*Fib.dMinor.*Fib.dMajor).^(1/3);

%% ------------------- Fibre Positioning ----------------------------- %

% Distance between Fibres
zDist=-(Fib.zmax-Fib.zmin)/(Fib.nFibZ-1);
Fib.PosXAll=[]; Fib.PosYAll=[];Fib.PosZAll=[];

for j=1:1:Fib.nAtomType
    for i=1:1:Fib.nFibZ
        Fib.Class(j).xPos(i,1)=Fib.x;       % x-postion
        Fib.Class(j).yPos(i,1)=Fib.y;       % y-postion
        Fib.z=Fib.zmax+(i-1)*zDist;         % z-postion
        Fib.Class(j).zPos(i,1)=Fib.z;

        Fib.Class(j).r(i,1)=(Fib.z^2+Fib.y^2)^0.5;
        Fib.Class(j).theta(i,1)=atan(Fib.y/Fib.z);
        if Fib.z==0
            Fib.Class(j).theta(i,1)=0;
        end
    end

    % All Position
    Fib.PosXAll=horzcat(Fib.PosXAll, Fib.Class(j).xPos(:,1));
    Fib.PosYAll=horzcat(Fib.PosYAll, Fib.Class(j).yPos(:,1));
    Fib.PosZAll=horzcat(Fib.PosZAll, Fib.Class(j).zPos(:,1));

end

%% ------------------- Fibre Velocity --------------------------------- %
```

```
% set Inlet Velocity in x direction acc. to Tamayol, Bahrami - Laminar Flow
% in Mircochannels With Noncircular Cross Section
% Set initial velocity

if Fib.setVeloX == 1
   for i=1:1:Fib.nAtomType
   r = sqrt(Fib.PosYAll(:,i).^2 + Fib.PosZAll(:,i).^2);
   theta = acos(Fib.PosYAll(:,i)./r);
   tempVeloX = velSCS(halfLength, Umax, r, theta);
   Fib.veloX(:,i)=tempVeloX;
     for ii=1:1:size(Fib.veloX(:,i),1);
       % Elimiate NAN velue at center of Channel
        if and(min(Fib.PosYAll(ii,i))==0,min(Fib.PosZAll(ii,i))==0);
           Fib.veloX(ii,i)=Umax;
        end

        % Elimiate negative Values
         if Fib.veloX(ii,i) < 0;
            Fib.veloX(ii,i) =0;
         end
     end
     plot3(Fib.PosYAll(:,1),Fib.PosZAll(:,1),Fib.veloX(:,1),'.');
     hold on;
   end
end
hold off;

%% ----------------------- Jeffery Orbit -------------------------------- %
Fib=func_JefferyOrbit(Fib);

%% -------------------- Write Output File ------------------------------ %
%% Create txt - File
% create_atoms    1 single initialPosX initialPosY1 initialPosZ1 units box
   ID =1;
for Type=1:1:Fib.nAtomType
 % Open output file
 InitFibPos=fopen(...
  fullfile(resDir,['in.ParticlePos_AR',num2str(Fib.AR(Type)),'.txt']),'w');
   for i=1:1:size(Fib.PosXAll,1);
      if or(isnan(Fib.PosYAll(i,Type)),isnan(Fib.PosZAll(i,Type)))
      else
        if Fib.setVeloX == 1 % Output with set velocities
        fprintf(InitFibPos,...
              '%s %i %s %5.4f %5.4f %5.4f %s \n %s %i %s %5.4f \n',...
              'create_atoms    ',Type+(Fib.startType-1),... % create atom
              'single ', Fib.PosXAll(i,Type),...
               Fib.PosYAll(i,Type),...
               Fib.PosZAll(i,Type),...
              ' units box ',...
              ' set atom ',...  % set atom ID vx veloX
              ID,...
              ' vx ',...
              Fib.veloX(i,Type));
          ID =ID+1;

          else % Output without set velocities
               fprintf(InitFibPos, '%s %i %s %5.4f %5.4f %5.4f %s \n',...
              'create_atoms    ',Type+(Fib.startType-1),... % create atom
              'single ', Fib.PosXAll(i,Type),...
               Fib.PosYAll(i,Type),...
               Fib.PosZAll(i,Type),...
              ' units box ');
          end
      end
   end
end
disp('.txt-File is done!')

%% -------------------- Plot Fibers at initial position ----------------- %
funcStat=0;
funcStat=func_PlotInitialPos(Fib.nAtomType,Fib,Box);

%% ---------------------------- End of Program -------------------------- %
disp('End of program - Have fun...')

```

```matlab
% Jeffery Orbit Calculation and plot
function Fib=func_JefferyOrbit(Fib)

global interpreterName resDir currDir figVisible halfLength Umax printAs

%% -------------------- Formating ----------------------------------------%
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

%% ------------------ Jeffery Orbit Calculations ------------------------%
for j=1:1:Fib.nAtomType
    for i=1:1:Fib.nFibZ

   % Velocity
   Fib.Class(j).veloX(i,1)=velSCS(halfLength, Umax,...
     Fib.Class(j).r(i,1), Fib.Class(j).theta(i,1));

   % Shear Rate
   Fib.Class(j).ShearRate(i,1) = ShearRateSCS(halfLength, Umax,...
     Fib.Class(j).r(i,1), Fib.Class(j).theta(i,1));

   % Orbit Period
   Fib.Class(j).tOrbit(i,1)=...
    2*pi.*(Fib.AR(j)+1./Fib.AR(j))./Fib.Class(j).ShearRate(i,1);

   % Half Orbit (pi = 180)
    Fib.Class(j).tOrbitHalf(i,1)=Fib.Class(j).tOrbit(i,1)*0.5;

   % Length for Orbit
    Fib.Class(j).sOrbitHalf(i,1)=...
     Fib.Class(j).veloX(i,1)*Fib.Class(j).tOrbitHalf(i,1);
    end

end

%% --------------------------- Figure / Plot  ----------------------------%
% ------------------ Length of Traveling for 0.5 orbit ------------------
figName = 's0.5Obrit_zInit';
scrs = get(0,'screensize');
rect = [10, 10, 600, 600]; %[left, bottom, width, height]
fig1 = figure('Name',figName,'Position',rect,'Visible',figVisible);
legendText={};

for j=1:1:Fib.nAtomType
    x=Fib.Class(j).zPos(:,1);
    y=Fib.Class(j).sOrbitHalf(:,1);

    plot(x,y,symbolArrayCS{j});hold on;

    legendText{j}=...
        [preAfterSymbol,'AR=',num2str(Fib.AR(j)),preAfterSymbol];
end

legend(legendText,'Location','Northwest');

 if(  strcmp(interpreterName, 'latex') )
   legend(legendText,'fontsize',fontSizeLabel,...
       'interpreter',interpreterName,'Location','Northwest');
 end

xlabel([preAfterSymbol,'z^{+}',preAfterSymbol],...
    'interpreter',interpreterName)
ylabel([preAfterSymbol,'s^{+}_{0.5 orbit}',preAfterSymbol],...
    'interpreter',interpreterName)

    grid off
    box on
    legend boxoff
    set(gca,'XMinorTick','on','YMinorTick','on')
    set(0,'defaultaxesfontsize',defaultaxesfontsize);
    set(0,'defaulttextfontsize',defaulttextfontsize);
    set(gca,'FontSize',fontSizeAxis);
    set(gca,'FontWeight','normal');
    xlhand = get(gca,'xlabel');
    ylhand = get(gca,'ylabel');
    zlhand = get(gca,'zlabel');
    set(xlhand,'fontsize',fontSizeLabel);
    set(ylhand,'fontsize',fontSizeLabel);
    set(zlhand,'fontsize',fontSizeLabel)
    set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
    set(xlhand,'FontWeight','bold');
    set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
    set(ylhand,'FontWeight','bold');
    set(zlhand,'FontName','Times'); set(zlhand,'FontAngle','italic');
    set(zlhand,'FontWeight','bold');

% Save Figure
Name='Particle_InitialPosition_sHalfOrbit';
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 20])
print(printAs,'-r450', [resDir,Name])

% -------------------------- Time for 0.4 orbit -------------------------
figName = 't0.5Obrit_zInit';
scrs = get(0,'screensize');
rect = [10, 10, 600, 600]; %[left, bottom, width, height]

fig2 = figure('Name',figName,'Position',rect,'Visible',figVisible);

legendText={};

for j=1:1:Fib.nAtomType
    x=Fib.Class(j).zPos(:,1);
    y=Fib.Class(j).tOrbitHalf(:,1);

    plot(x,y,symbolArrayCS{j});hold on;

    legendText{j}=...
        [preAfterSymbol,'AR=',num2str(Fib.AR(j)),preAfterSymbol];
end

legend(legendText,'Location','Northwest');

 if(  strcmp(interpreterName, 'latex') )
   legend(legendText,'fontsize',fontSizeLabel,...
       'interpreter',interpreterName,'Location','Northwest');
 end


xlabel([preAfterSymbol,'z^{+}',preAfterSymbol],...
    'interpreter',interpreterName)
ylabel([preAfterSymbol,'t^{+}_{0.5 orbit}',preAfterSymbol],...
    'interpreter',interpreterName)

    grid off
    box on
    legend boxoff
    set(gca,'XMinorTick','on','YMinorTick','on')
    set(0,'defaultaxesfontsize',defaultaxesfontsize);
    set(0,'defaulttextfontsize',defaulttextfontsize);
    set(gca,'FontSize',fontSizeAxis);
    set(gca,'FontWeight','normal');
    xlhand = get(gca,'xlabel');
    ylhand = get(gca,'ylabel');
    zlhand = get(gca,'zlabel');
    set(xlhand,'fontsize',fontSizeLabel);
    set(ylhand,'fontsize',fontSizeLabel);
    set(zlhand,'fontsize',fontSizeLabel)
    set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
```

```
function funcStat=func_PlotInitialPos(nAtomType,Fib,Box);

% Plot initial positions of all fibers in channel with square cross section

global interpreterName resDir figVisible printAs

%% -------------------- Formating ----------------------------------------%
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

%% --------------------  Figure / Plot  ----------------------------------%
% -------------------- Plot yz-Plane ---------------------------------%
figName = 'Initial position of all fibers yz';
scrs = get(0,'screensize');
rect = [10, 10, 600, 600]; %[left, bottom, width, height]

fig1 = figure('Name',figName,'Position',rect,'Visible',figVisible);

% subplot(2,1,1) % yz
for Type=1:1:nAtomType
          plot(Fib.PosYAll(:,Type), Fib.PosZAll(:,Type),...
           symbolArrayCS{Type},...
          'MarkerSize',markersize*Type*0.1);hold on;
     legendText{Type}=...
          [preAfterSymbol,'AR=',num2str(Fib.AR(Type)),preAfterSymbol];
   end

 legend(legendText,'Location','NorthEast');

 if(  strcmp(interpreterName, 'latex') )
   legend(legendText,'fontsize',fontSizeLabel,...
      'interpreter',interpreterName,'Location','NorthEast');
 end

   % Lines
   % plot([Box.yMin Box.yMax],[0 0],'k-.','MarkerSize',2);hold on;
   % plot([0 0],[Box.zMin Box.zMax],'k-.','MarkerSize',2);hold off;

   axis([Box.yMin Box.yMax Box.zMin Box.zMax]);

   set(gca,'YTick',[Box.zMin:(Box.zMax-Box.zMin)/4:Box.zMax]);
   set(gca,'XTick',[Box.yMin:(Box.yMax-Box.yMin)/4:Box.yMax]);
%     title([preAfterSymbol,'initial fiber distribution',' y^{+}-z^{+}',...
%          preAfterSymbol],...
%         'Interpreter',interpreterName,'fontsize',fontSizeTitle);
   xlabel([preAfterSymbol,'y^{+}',preAfterSymbol],...
       'Interpreter',interpreterName);
   ylabel([preAfterSymbol,'z^{+}',preAfterSymbol],...
       'Interpreter',interpreterName);

   grid off
   box on
   legend boxoff
   set(gca,'XMinorTick','on','YMinorTick','on')
   set(0,'defaultaxesfontsize',defaultaxesfontsize);
   set(0,'defaulttextfontsize',defaulttextfontsize);
   set(gca,'FontSize',fontSizeAxis);
   set(gca,'FontWeight','normal');
   xlhand = get(gca,'xlabel');
   ylhand = get(gca,'ylabel');
   zlhand = get(gca,'zlabel');
   set(xlhand,'fontsize',fontSizeLabel);
   set(ylhand,'fontsize',fontSizeLabel);
   set(zlhand,'fontsize',fontSizeLabel)
   set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
   set(xlhand,'FontWeight','bold');
   set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
   set(ylhand,'FontWeight','bold');
```

```
   set(xlhand,'FontWeight','bold');
   set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
   set(ylhand,'FontWeight','bold');
   set(zlhand,'FontName','Times'); set(zlhand,'FontAngle','italic');
   set(zlhand,'FontWeight','bold');

%% -------------------- Save Figure ---------------------------------%
Name='Particle_InitialPosition_tHalfOrbit';

set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 20])
print(printAs,'-r450', [resDir,Name])



end

```

```
   set(zlhand,'FontName','Times'); set(zlhand,'FontAngle','italic');
   set(zlhand,'FontWeight','bold');

% -------------------- Save Figure ----------
Name='Particle_InitialPosition_yz';
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 20])
print(printAs,'-r450', [resDir,Name])

% -------------------- Plot xy-Plane ---------------------------------%
figName = 'Initial position of all fibers xy';
scrs = get(0,'screensize');
rect = [10, 10, 600, 600]; %[left, bottom, width, height]

fig1 = figure('Name',figName,'Position',rect,'Visible',figVisible);
% subplot(2,1,2) % xy
   for Type=nAtomType:-1:1
        plot(Fib.PosXAll(:,Type), Fib.PosYAll(:,Type),...
          symbolArrayCS{Type},...
         'MarkerSize',markersize*0.25);hold on;
   end
   axis([Box.xMin Box.xMax Box.zMin Box.zMax])
   axis square
%     title([preAfterSymbol,'initial fiber distribution','x^{+}-y^{+}',...
%          preAfterSymbol],...
%         'Interpreter',interpreterName,'fontsize',fontSizeTitle);
   xlabel([preAfterSymbol,'x^{+}',preAfterSymbol],...
       'Interpreter',interpreterName);
   ylabel([preAfterSymbol,'y^{+}',preAfterSymbol],...
       'Interpreter',interpreterName);

    grid off
    box on
    legend boxoff
    set(gca,'XMinorTick','on','YMinorTick','on')
    set(0,'defaultaxesfontsize',defaultaxesfontsize);
    set(0,'defaulttextfontsize',defaulttextfontsize);
    set(gca,'FontSize',fontSizeAxis);
    set(gca,'FontWeight','normal');
    xlhand = get(gca,'xlabel');
    ylhand = get(gca,'ylabel');
    zlhand = get(gca,'zlabel');
    set(xlhand,'fontsize',fontSizeLabel);
    set(ylhand,'fontsize',fontSizeLabel);
    set(zlhand,'fontsize',fontSizeLabel)
    set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
    set(xlhand,'FontWeight','bold');
    set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
    set(ylhand,'FontWeight','bold');
    set(zlhand,'FontName','Times'); set(zlhand,'FontAngle','italic');
    set(zlhand,'FontWeight','bold');

% -------------------- Save Figure ----------
Name='Particle_InitialPosition_xy';
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 20])
print(printAs,'-r450', [resDir,Name])

funcStat=1;
end

```

```
function [ShearRateSqu] = ShearRateSCS(s, Umax, r, theta)

% INPUT
%    m    ...number of sides (4...square)
%    s    ...half side length
%    Umax ...maximal velocity
%    r    ...radial distance
%    theta...angular position of the point
% OUTPUT
% ShearRateSqu  ...the shear rate at the point (r, theta)
% ShearRate SQU = ((du/dr)^2+(du/dtheta)^2)^0.5
%
% Source: S.Radl - Direct Numerical Simulation of Reactive
% Deformable Bubbles in non-Newtonian FluidsDiploma-Thesis
m=4;

A1=0.247+0.767/m^2;
C(1)=(6.01-3.12*m-0.965*m^2)^-1;
if(m==4)
    C(2)=-1.096e-3;
end

eta = r .* tan(pi/m) ./ s;
dudr     = zeros(size(eta,1),size(eta,2));
dudtheta = zeros(size(eta,1),size(eta,2));
for i=1:size(eta,1)
    for j=1:size(eta,2)
        currr = r(i,j);
        currTheta=theta(i,j);
        dudr(i,j)     = -2.*currr.*(tan(pi/m) ./ s)^2./4./A1;
        dudtheta(i,j) = 0;
        for it=1:length(C)
            dudr(i,j)    = dudr(i,j)+ C(it)/A1*m*it.*currr^(m*it-1)*...
                               (tan(pi/m)/s)^(m*it)*cos(m*currTheta);
            dudtheta(i,j)= dudtheta(i,j)+(-C(it)/A1.*currr.^(m*it)*...
                               (tan(pi/m)/s)^(m*it)*m*sin(m*currTheta));
        end
    end
end

dudr = dudr.*Umax;
dudtheta = dudtheta.*Umax;
ShearRateSqu = (dudr.^2+dudtheta.^2).^0.5;
end
```

```
function [vel] = velSCS(s, UmaxSqu, r, theta)
% velDistributionPolygonialDuct - calculates the velocity distribution in a
% polygonial duct. Velocity scaled to give a mean velocity Umean
% Approach based on Tamayol and Bahrami, J Fluids Engineering 132, 111201
% [vel] = velDistributionPolygonialDuct(m, s, Umax, r, theta)
% INPUT
%    m    ...number of sides (4...square)
%    s    ...half side length
%    Umax ...maximal velocity
%    r    ...radial distance
%    theta...angular position of the point
% OUTPUT
%    vel  ...the velocity at the point (r, theta)

m=4;

A1=0.247+0.767/m^2;
C(1)=(6.01-3.12*m-0.965*m^2)^-1;
if(m==4)
    C(2)=-1.096e-3; % S. Radl
end

eta = r .* tan(pi/m) ./ s;
uStar = zeros(size(eta,1),size(eta,2));
for i=1:size(eta,1)
    for j=1:size(eta,2)
        currEta=eta(i,j);
        currTheta=theta(i,j);
        uStar(i,j) = 1-currEta^2/4/A1;
        for it=1:length(C)
            uStar(i,j)=uStar(i,j)+C(it)/A1*currEta^(m*it)*cos(m*currTheta);
        end
    end
end

vel = UmaxSqu*uStar;
end
```

```
% Program to calculate Fibre Orbits in SCS Channel
% SCS .. square cross section
% (c) Lisa Koenig
%
% Required dump:
% 1          ... id
% 2          ... type
% 3 4 5       ... x y z
% 6 7 8       ... vx vy vz
% 9 10 11     ... fx fy fz
% 12 13 14    ...omegax omegay omegaz
% 15          ... radius
% 16 17 18    ... f_ex[1] f_ex[2] f_ex[3]
% 19          ... f_shape[1]
% 20 21 22 23 ...c_cQuatw c_cQuati c_cQuatj c_cQuatk

clear all; close all; clc
global inDir resDir homeDir

%% ---------------------- Input ---------------------------
Umax=2.08;  % max. velocity in channel
s=0.5;      % half Height/Width of channel
dMajor=[0.04 0.1 0.2 0.3 0.4]; % dMajor values of the classes smallest to largest

inDir='../../post';
resDir='../../postprecessing';
homeDir=pwd;

%% -------- Analytical results using Jeffery orbit ----------

% dMinor and Aspect Ratio of the rolate Spheriod
dMinor=ones(1,length(dMajor))*0.02;
AR=dMajor./dMinor;

% max. position of the fibre class requiring a 90 degree spin
r=s-dMajor/2;
theta=ones(1,length(r))*pi/2;

% SCS laminar velocity profile
vel = velSCS(s, Umax, r, theta);

% Rate of the Strain Tensor
ShearRate = ShearRateSCS(s, Umax, r, theta);

% Orbit Period
tOrbit=2*pi.*(AR+1./AR)./ShearRate
rotRate=ShearRate./(AR+1./AR);

% Quater Orbit (pi = 180)
tOrbitHalf=tOrbit./2

% Length for Orbit
sOrbitHalf=vel.*tOrbitHalf
sOrbit=vel.*tOrbit;

%plot(r,sOrbitqu,'o')
%xlabel('r')
%ylabel('s_{pi/2}')

%% -------------  Output Analytical Results --------------------------- %
outFile=fopen('../../postProcessing/FibreOrbits.txt','w');
fprintf(outFile,'%s \n','# Theoretical Jeffery orbit results');
fprintf(outFile,'%s \n',...
'# dMajor r_max velo ShearRate HalfOrbitTime TravelLengthHalfOrbit');
for i=1:1:length(dMajor)

    fprintf(outFile,...
        '%5.3f \t %5.3f \t %5.3f \t %5.3f \t %5.3f \t %5.3f  \n',...
        dMajor(i), r(i), vel(i), ShearRate(i), tOrbitHalf(i),sOrbitHalf(i));
end
```

```
%% This preprocessing program provides Fibre Data (position and velocity)
% (e.g for usage in a CFDEM simulation)
% Lisa Koenig, August 2015, TUGraz
%
%  Schematic Diagramm:
% ---------------------
%
%           Z-dir.                        Z-dir.
%             /\                           /\
%             ||                           ||
%             ||                           ||
%          +--------+    +--------------------------------+
%          |        |    |                                |
%  Y-dir.<- |        |    |                                | -> X-dir.
%          |        |    |                                |
%          +--------+    +------------+---------+----------+
%                              |       |
%                              |       |
%
% The fibres are positioned on a yz-plane perpendicular to the
% flow direction (x-direction).The fibres can either be spread over:
% (partOfCS=1) the whole corss section,
% (partOfCS=2) half of the cross section,
% (partOfCS=41) in the lower right quater,
% (partOfCS=42) in the upper right quater of the corss section.
%
% Multiple Fibre classes (separated acc. to thei major length, dMajor)
% can be inserted. The positioning of the fibres, which is always limited
% by the required distance to the wall(dMajor/2), since the fibres are
% not allowed to intersect the wall, can be done in 3 different ways
% (0) Distance between fibres in each class is based on each classes
%     dMajor, number of fibres in each class is the same.
% (1) Distance of all Fibres in all classes is given by shortest
%     fibre (class), fibres that don't fit because they are to long
%     and thus to close to the wall are not set. Fibre positions
%     are the same for all fibres. It is just possible that on the
%     wall-near positions there aren't all fibres present.
% (2) Distance of all Fibres in all classes is given by longest fibre
%     (class). Thus all fibre stypes are on all positions.
%     Distance to the wall is given bythe longest fibre.
%     -> Short fibres that would fit close to the wall are still far way.
%
% The Fibre initial velocity can be set with the Velocity profil given by:
% Tamayol, Bahrami - Laminar Flow in Mircochannels With
% Noncircular Cross Section
% setVeloX=0 ... initial fibre velocity is zero
% setVeloX=1 ... initial fibre velocity is calculated

clear all; clc; close all; fclose all
global interpreterName resDir currDir

%% -------------------- Directories ------------------------------------- %
inDir = '../../input/';
fileName= 'input_Case_1.txt';
resDir='../../preProcessing/';
currDir=pwd;

%% -------------------- Read temporary file for setting ----------------- %
% Check if octave of matlab is used
tempDataFile = fopen('tempData.txt');
C = textscan(tempDataFile,'%f %f',1);
fclose(tempDataFile);
InterPreterName=C{1,1};

if InterPreterName == 1 % MATLAB
    interpreterName='latex'
end

if InterPreterName == 2 %OCTAVE
    interpreterName='tex'

%% Read Energy Data
% Check number of dump-files in folder
cd(inDir);
file=dir('energy.dat'); % list all dump-files
cd(homeDir);

% Read energy Data file: time eKin eRot ePot eTot
Data=dlmread(fullfile(inDir,file.name),'',1,0);

%% Plot results
% Numerical results (rotational energy)
plot(Data(:,1),Data(:,3)); hold on;

% Jeffery Orbit
for i=1:1:length(dMajor)
plot([tOrbitHalf(i),tOrbitHalf(i)], [0, max(Data(:,3))]); hold on;
end

xlabel('t^{+}')
ylabel('E_{rot}')
```

```
end

%% -------------------- Prepare environment ----------------------------- %
% Check if output directory exists
cd ..
cd ..
if exist('preProcessing') == 0 % if folder does not exist create it!
   mkdir('preProcessing')
end
cd (currDir)

%% -------------------- Read Input Data --------------------------------- %

    % Get Data from input File via dlmread()
    temp.Data1 = dlmread(strcat(inDir,fileName),'',[18 0 18 10]);
    temp.Data2 = dlmread(strcat(inDir,fileName),'',[23 0 23 1]);
    temp.Data3 = dlmread(strcat(inDir,fileName),'',[26 0 26 1]);
    temp.Data4 = dlmread(strcat(inDir,fileName),'',29,0);

    % Box Dimensions and Calculation Option
    Box.xMin = temp.Data1(1);
    Box.xMax = temp.Data1(2);
    Box.yMin = temp.Data1(3);
    Box.yMax = temp.Data1(4);
    Box.zMin = temp.Data1(5);
    Box.zMax = temp.Data1(6);
    Box.xPosAll= temp.Data1(7);
    Box.partOfCS = temp.Data1(8);
    Box.sameDist = temp.Data1(9);
    Fib.startType = temp.Data1(10);
    Fib.setVeloX = temp.Data1(11);

    % Fibre/Fluid Data
    Fib.rho  = temp.Data2(1);
    rhoFluid = temp.Data2(2);

    % Number of Fibres in Z and Y direction
    Fib.nFibY = temp.Data3(1);
    Fib.nFibZ = temp.Data3(2);

    % Fibre Data
    Fib.Data = sortrows(temp.Data4,[1 2]); % Sort acc. to dMajor
    Fib.dMajor = Fib.Data(:,1);
    Fib.dMinor = Fib.Data(:,2);

    clear temp.*; % clear temporary data


%% -------------------- Fibre Values Calculation ------------------------ %
% Diameter of Spherocylinder
Fib.dCylinder =1.24.*Fib.dMinor./sqrt(log(Fib.dMajor./Fib.dMinor));
% Aspect Ratios
Fib.AR        =Fib.dMajor./Fib.dMinor;
% Vol. single fibre in class i; fibre assumed to be ellipsoid
Vol.Fibi      = Fib.dMajor.^3 ./ Fib.AR .^2 .* pi ./ 6;
% mass of one single fibre in class i
mass.Fibi     = Fib.rho * Vol.Fibi;
% vol. equiv. diameter
Fib.dSphere   = (Fib.dMinor.*Fib.dMinor.*Fib.dMajor).^(1/3);

%% -------------------- Fibre Positioning ------------------------------ %
% number of atom types for output file -> CFDEM
nAtomType = length(Fib.AR);

% Number of Fibres depending on the choosen cross section area
    if Box.partOfCS == 1;    % total cross section
        disp('Total cross section');
    elseif Box.partOfCS == 2; % half
        %  Fib.nFibY = Fib.nFibY/2;
        disp('Half cross section');
    elseif or(Box.partOfCS == 41,Box.partOfCS == 42); % quater
        %  Fib.nFibY=Fib.nFibY/2;
        %  Fib.nFibZ=Fib.nFibZ/2;
        disp('Quater cross section');
    else
        error('Box.partOfCS has to be 1, 2, 41, 42');
    end
    Fib.nFibTot=Fib.nFibZ*Fib.nFibY;

% set max./min. y- and z-position for all fibers
% depending on the choosen cross section area
  if Box.partOfCS == 1;    % total cross section
   yMinPos = Box.yMin+Fib.dMajor/2;  yMaxPos = Box.yMax-Fib.dMajor/2;
   zMinPos = Box.zMin+Fib.dMajor/2;  zMaxPos = Box.zMax-Fib.dMajor/2;

  elseif Box.partOfCS == 2; % half the cross section
   yMinPos = ones(length(Fib.dMajor),1)*0; yMaxPos = Box.yMax-Fib.dMajor/2;
   zMinPos = Box.zMin+Fib.dMajor/2;  zMaxPos = Box.zMax-Fib.dMajor/2;

  elseif Box.partOfCS == 41; % high quater of the cross section
   yMinPos = ones(length(Fib.dMajor),1)*0; yMaxPos = Box.yMax-Fib.dMajor/2;
   zMinPos = Box.zMin+Fib.dMajor/2; zMaxPos = ones(length(Fib.dMajor),1)*0;

  elseif Box.partOfCS == 42; % low quater of the cross section
   yMinPos = ones(length(Fib.dMajor),1)*0; yMaxPos = Box.yMax-Fib.dMajor/2;
   zMinPos = ones(length(Fib.dMajor),1)*0; zMaxPos = Box.zMax-Fib.dMajor/2;
    end

  % Fibre Distance in y -direction for all classes
     FibDistY = (abs(yMaxPos-yMinPos))./(Fib.nFibY-1);
  % Fibre Distance in z -direction for all classes
     FibDistZ = (abs(zMaxPos-zMinPos))./(Fib.nFibZ-1);

% Loop over all Fibre Types
for Type=1:1:nAtomType

    % Switch: How are the fibres distributed
    if Box.sameDist == 0     % 0...all classes dist acc. to length
        s = Type;
    elseif Box.sameDist == 1 % 1...all Fib same dist based on shortest fibs
        s = 1;
    elseif Box.sameDist == 2 % 2...all Fib same dist based on longest fib
        s = nAtomType;
    else
        disp('Check sameDist! it can eiter be 0, 1 or 2')
    end

   % Possible Length Intervall in z and y direction
   InterY = [yMinPos(s) yMaxPos(s)];
   InterZ = [zMinPos(s) zMaxPos(s)];

      % Fibres y-directions
      for i=1:1:Fib.nFibY;
       tempPosY= InterY(1)+ FibDistY(s).*(i-1);
          if and(tempPosY >= Box.yMin+Fib.dMajor(Type)/2,...
                 tempPosY <= Box.yMax-Fib.dMajor(Type)/2)
             Fib.PosY(i,Type)= tempPosY;
          else
             Fib.PosY(i,Type)=NaN;
          end
      end

      % Fibres z-directions
       for i=1:1:Fib.nFibZ;
       tempPosZ= InterZ(1)+ FibDistZ(s).*(i-1);
          if and(tempPosZ >= Box.zMin+Fib.dMajor(Type)/2,...
                 tempPosZ <= Box.zMax-Fib.dMajor(Type)/2);
             Fib.PosZ(i,Type)= tempPosZ;
          else
             Fib.PosZ(i,Type)=NaN;
```

```matlab
      end
    end

    % Combine Positions
    for i=1:1:length(Fib.PosY(:,Type))
      for j=1:1:length(Fib.PosZ(:,Type))
      Fib.PosYAll(i,j,Type) = Fib.PosY(i,Type);
      Fib.PosZAll(i,j,Type) = Fib.PosZ(j,Type);
      end
    end

    % Fibre x-directions
    Fib.PosXAll=Fib.PosYAll;
    Fib.PosXAll(:,:,:)=Box.xPosAll;

end

%% -------------------- Set Fibre Velocity ----------------------------- %
% set Inlet Velocity in x direction acc. to Tamayol, Bahrami - Laminar Flow
% in Mircochannels With Noncircular Cross Section
% Set initial velocity
halfLength=0.5;
UmaxSqu=0.5;

if Fib.setVeloX == 1
  for i=1:1:nAtomType
  r = sqrt(Fib.PosYAll(:,:,i).^2 + Fib.PosZAll(:,:,i).^2);
  theta = acos(Fib.PosYAll(:,:,i)./r);
  tempVeloX = velSCS(halfLength, UmaxSqu, r, theta);
  Fib.veloX(:,:,i)=tempVeloX

   for ii=1:1:size(Fib.veloX(:,:,i),1)
    for jj=1:1:size(Fib.veloX(:,:,i),2)

    % Elimiate NAN velue at center of Channel
     if and(min(Fib.PosYAll(ii,jj,i))==0,min(Fib.PosZAll(ii,jj,i))==0)
       Fib.veloX(ii,jj,i)=UmaxSqu;
     end

     % Elimiate negative Values
     if Fib.veloX(ii,jj,i) < 0
        Fib.veloX(ii,jj,i) =0;
     end
    end
   end
   plot3(Fib.PosYAll(:,:,1),Fib.PosZAll(:,:,1),Fib.veloX(:,:,1),'.');
   hold on;
  end
end

%% -------------------- Write Output File ------------------------------ %
%% Create txt - File
% create_atoms    1 single initialPosX initialPosY1 initialPosZ1 units box
  ID =1;
for Type=1:1:nAtomType
 % Open output file
 InitFibPos=fopen(...
  fullfile(resDir,['in.ParticlePos_AR',num2str(Fib.AR(Type)),'.txt']),'w');

  for i=1:1:size(Fib.PosXAll,1);
   for j=1:1:size(Fib.PosXAll,2)

    if or(isnan(Fib.PosYAll(i,j,Type)),isnan(Fib.PosZAll(i,j,Type)))
    else
     if Fib.setVeloX == 1 % Output with set velocities
     fprintf(InitFibPos,...
        '%s %i %s %5.4f %5.4f %5.4f %s \n %s %i %s %5.4f \n',...
        'create_atoms    ',Type+(Fib.startType-1),... % create atom
        'single ', Fib.PosXAll(i,j,Type),...
         Fib.PosYAll(i,j,Type),...
         Fib.PosZAll(i,j,Type),...
        ' units box ',...
        ' set atom ',...  % set atom ID vx veloX
        ID,...
        ' vx ',...
        Fib.veloX(i,j,Type));
      ID =ID+1;

      else % Output without set velocities
         fprintf(InitFibPos, '%s %i %s %5.4f %5.4f %5.4f %s \n',...
        'create_atoms    ',Type+(Fib.startType-1),... % create atom
        'single ', Fib.PosXAll(i,j,Type),...
         Fib.PosYAll(i,j,Type),...
         Fib.PosZAll(i,j,Type),...
        ' units box ');
      end
    end
   end
  end
end
disp('.txt-File is done!')

%% -------------------- Plot Fibers at initial position ---------------- %
funcStat=0;
funcStat=func_PlotInitialPos(nAtomType,Fib,Box);

%% --------------------------- End of Program ------------------------- %
disp('End of program - Have fun...')
```

```matlab
function funcStat=func_PlotInitialPos(nAtomType,Fib,Box);

% Plot initial positions of all fibers in channel with square cross section

global interpreterName resDir

%% -------------------- Formating ----------------------------------------%
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

%% --------------------  Figure / Plot  ----------------------------------%
figName = 'Initial position of all fibers';
scrs = get(0,'screensize');
rect = [10, 10, 400, 800]; %[left, bottom, width, height]

fig1 = figure('Name',figName,'Position',rect);

% -------------------- SubPlot yz-Plane ---------------------------------%
subplot(2,1,1) % yz
    legendText='';
    for Type=nAtomType:-1:1
%         Type=5
      plot(Fib.PosYAll(:,:,Type), Fib.PosZAll(:,:,Type),...
          'ro','MarkerSize',1.5*(Type+1));
          %'color',[1*Type/nAtomType 1*Type/nAtomType 1*Type/nAtomType]);
      hold on;
      legendText{Type}=...
          [preAfterSymbol,'AR=',num2str(Fib.AR(Type)),preAfterSymbol];
    end

    % Lines
    plot([Box.yMin Box.yMax],[0 0],'k-.','MarkerSize',2);hold on;
    plot([0 0],[Box.zMin Box.zMax],'k-.','MarkerSize',2);hold on;

    legend(legendText,'Location','eastoutside');

 if(  strcmp(interpreterName, 'latex') )
   legend(legendText,'fontsize',fontSizeLabel,...
       'interpreter',interpreterName,'Location','NorthWest');
 end

    axis([Box.yMin Box.yMax Box.zMin Box.zMax]);
    axis square;
    set(gca,'ZLim',[Box.zMin Box.zMax]);
    set(gca,'YLim',[Box.yMin Box.yMax]);
    title('initial fiber distribution y-z',...
        'Interpreter',interpreterName,'fontsize',fontSizeTitle);
    xlabel([preAfterSymbol,'y',preAfterSymbol],...
        'Interpreter',interpreterName);
    ylabel([preAfterSymbol,'z',preAfterSymbol],...
        'Interpreter',interpreterName);

    grid off
    box on
    legend boxon
    set(0,'defaultaxesfontsize',defaultaxesfontsize);
    set(0,'defaulttextfontsize',defaulttextfontsize);
    set(gca,'FontSize',fontSizeAxis);
    set(gca,'FontWeight','normal');
    xlhand = get(gca,'xlabel');
    ylhand = get(gca,'ylabel');
    zlhand = get(gca,'zlabel');
    set(xlhand,'fontsize',fontSizeLabel);
    set(ylhand,'fontsize',fontSizeLabel);
    set(zlhand,'fontsize',fontSizeLabel)
    set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
    set(xlhand,'FontWeight','bold');
    set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
    set(ylhand,'FontWeight','bold');
    set(zlhand,'FontName','Times'); set(zlhand,'FontAngle','italic');
    set(zlhand,'FontWeight','bold');

% -------------------- SubPlot xy-Plane ---------------------------------%
subplot(2,1,2) % xy
    for Type=nAtomType:-1:1
       plot(Fib.PosXAll(:,:,Type), Fib.PosYAll(:,:,Type),'.',...
           'MarkerSize',2*Type);hold on;
    end
    axis([Box.xMin Box.xMax Box.zMin Box.zMax])
    axis equal
    set(gca,'xLim',[Box.xMin Box.xMax])
    set(gca,'YLim',[Box.zMin Box.zMax])
    title('initial fiber distribution x-y',...
        'Interpreter',interpreterName,'fontsize',fontSizeTitle);
    xlabel([preAfterSymbol,'x',preAfterSymbol],...
        'Interpreter',interpreterName);
    ylabel([preAfterSymbol,'y',preAfterSymbol],...
        'Interpreter',interpreterName);

    grid off
    box on
    legend boxoff
    set(0,'defaultaxesfontsize',defaultaxesfontsize);
    set(0,'defaulttextfontsize',defaulttextfontsize);
    set(gca,'FontSize',fontSizeAxis);
    set(gca,'FontWeight','normal');
    xlhand = get(gca,'xlabel');
     ylhand = get(gca,'ylabel');
     zlhand = get(gca,'zlabel');
     set(xlhand,'fontsize',fontSizeLabel);
     set(ylhand,'fontsize',fontSizeLabel);
     set(zlhand,'fontsize',fontSizeLabel)
     set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
     set(xlhand,'FontWeight','bold');
     set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
     set(ylhand,'FontWeight','bold');
     set(zlhand,'FontName','Times'); set(zlhand,'FontAngle','italic');
     set(zlhand,'FontWeight','bold');

%% -------------------- Save Figure ---------------------------------%
Name='Particle_InitialPosition';
printAs='-dpdf';
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 18])
print(printAs,'-r450', [resDir,Name])

funcStat=1;
end
```

```
Input Data:
=====================================================================
xMin xMax yMin yMax zMin zMax ... simulation domain
dMajor    ... dMajor of Fibre class
dMinor    ... dMinor of Fibre class
x,y       ... x,y position of all fibres
zmin,zmax ... min and max z poition
nFibZ     ... number of fibres in z-direction
rhoFib    ... density of Fibre
rhoFluid  ... density of fluid
setVelox  ... set velocity in x-direction
Umax      ... Max velocity in Channel
halfLength ... half channel height

xMin    xMax    yMin    yMax    zMin    zMax
 -2.5   6.5     -0.5    0.5     -0.5    0.5

rhoFib  rhoFluid nFibZ  setVeloX  Umax halfLength  x    y   zmin  zmax startType
 1.27    1.0       100      1      2.08      0.5  -0.01 0.0 -0.48  0.0    2

dMajor         dMinor
0.040          0.020
0.100          0.020
0.200          0.020
0.300          0.020
0.400          0.020

```

```
%% This preprocessing program provides Fibre Data (position and velocity)
% (e.g for usage in a CFDEM simulation)
% Lisa Koenig, August 2015, TUGraz
%
%  Schematic Diagramm:
% ---------------------
%
%           Z-dir.                    Z-dir.
%             /\                        /\
%             ||                        ||
%             ||                        ||
%          +--------+    +---------------------------------+
%          |        |    |                        |
%  Y-dir.<- |        |    |                        | -> X-dir.
%          |        |    |                        |
%          +--------+    +-----------+---------+----------+
%                             |       |
%                             |       |
%
% The Fibre initial velocity can be set with the Velocity profil given by:
% Tamayol, Bahrami - Laminar Flow in Mircochannels With
% Noncircular Cross Section
% setVeloX=0 ... initial fibre velocity is zero
% setVeloX=1 ... initial fibre velocity is calculated

clear all; clc; close all; fclose all
global interpreterName resDir currDir figVisible halfLength Umax

%% -------------------- Directories ------------------------------------ %
inDir = '../../input/';
fileName= 'input.txt';
resDir='../../preProcessing/';
currDir=pwd;

figVisible='on';

%% -------------------- Read temporary file for setting ----------------- %
% Check if octave of matlab is used
tempDataFile = fopen('tempData.txt');
C = textscan(tempDataFile,'%f %f',1);
fclose(tempDataFile);
InterPreterName=C{1,1};

if InterPreterName == 1 % MATLAB
   interpreterName='latex'
end

if InterPreterName == 2 %OCTAVE
   interpreterName='tex'
end

%% -------------------- Prepare environment ---------------------------- %
% Check if output directory exists
cd ..
cd ..
if exist('preProcessing') == 0 % if folder does not exist create it!
  mkdir('preProcessing');
end
cd (currDir)

%% -------------------- Read Input Data -------------------------------- %

   % Get Data from input File via dlmread()
   temp.Data1 = dlmread(strcat(inDir,fileName),'',[15 0 15 5]);
   temp.Data2 = dlmread(strcat(inDir,fileName),'',[18 0 18 10]);
   temp.Data3 = dlmread(strcat(inDir,fileName),'',21,0);

   % Box Dimensions
   Box.xMin     = temp.Data1(1);
   Box.xMax     = temp.Data1(2);
```

```matlab
  Box.yMin     = temp.Data1(3);
  Box.yMax     = temp.Data1(4);
  Box.zMin     = temp.Data1(5);
  Box.zMax     = temp.Data1(6);

  % Fibre/Fluid Data
  Fib.rho  = temp.Data2(1);
  rhoFluid = temp.Data2(2);

  % Set velocity
  Fib.nFibZ    = temp.Data2(3);
  Fib.setVeloX = temp.Data2(4);
  Umax         = temp.Data2(5);
  halfLength   = temp.Data2(6);
  Fib.x        = temp.Data2(7);
  Fib.y        = temp.Data2(8);
  Fib.zmin     = temp.Data2(9);
  Fib.zmax     = temp.Data2(10);
  Fib.startType= temp.Data2(11);

  % Fibre Data
  Fib.Data = sortrows(temp.Data3,[1 2]); % Sort acc. to dMajor
  Fib.dMajor = Fib.Data(:,1);
  Fib.dMinor = Fib.Data(:,2);

  clear temp.*; % clear temporary data


%% -------------------- Fibre Values Calculation ------------------------ %
% Fibre Classes
Fib.nAtomType = length(Fib.dMajor);
% Diameter of Spherocylinder
Fib.dCylinder =1.24.*Fib.dMinor./sqrt(log(Fib.dMajor./Fib.dMinor));
% Aspect Ratios
Fib.AR        =Fib.dMajor./Fib.dMinor;
% Vol. single fibre in class i; fibre assumed to be ellipsoid
Vol.Fibi      = Fib.dMajor.^3 ./ Fib.AR .^2 .* pi ./ 6;
% mass of one single fibre in class i
mass.Fibi     = Fib.rho * Vol.Fibi;
% vol. equiv. diameter
Fib.dSphere   = (Fib.dMinor.*Fib.dMinor.*Fib.dMajor).^(1/3);

%% -------------------- Fibre Positioning ------------------------------ %

% Distance between Fibres
zDist=-(Fib.zmax-Fib.zmin)/(Fib.nFibZ-1);
Fib.PosXAll=[]; Fib.PosYAll=[];Fib.PosZAll=[];

for j=1:1:Fib.nAtomType
    for i=1:1:Fib.nFibZ
        Fib.Class(j).xPos(i,1)=Fib.x;       % x-postion
        Fib.Class(j).yPos(i,1)=Fib.y;       % y-postion
        Fib.z=Fib.zmax+(i-1)*zDist;         % z-postion
        Fib.Class(j).zPos(i,1)=Fib.z;

        Fib.Class(j).r(i,1)=(Fib.z^2+Fib.y^2)^0.5;
        Fib.Class(j).theta(i,1)=atan(Fib.y/Fib.z);
        if Fib.z==0
            Fib.Class(j).theta(i,1)=0;
        end
    end

    % All Position
    Fib.PosXAll=horzcat(Fib.PosXAll, Fib.Class(j).xPos(:,1));
    Fib.PosYAll=horzcat(Fib.PosYAll, Fib.Class(j).yPos(:,1));
    Fib.PosZAll=horzcat(Fib.PosZAll, Fib.Class(j).zPos(:,1));

end

%% ------------------- Fibre Velocity ---------------------------------- %
% set Inlet Velocity in x direction acc. to Tamayol, Bahrami - Laminar Flow
% in Mircochannels With Noncircular Cross Section
% Set initial velocity

if Fib.setVeloX == 1
    for i=1:1:Fib.nAtomType
    r = sqrt(Fib.PosYAll(:,i).^2 + Fib.PosZAll(:,i).^2);
    theta = acos(Fib.PosYAll(:,i)./r);
    tempVeloX = velSCS(halfLength, Umax, r, theta);
    Fib.veloX(:,i)=tempVeloX;
      for ii=1:1:size(Fib.veloX(:,i),1);
        % Elimiate NAN velue at center of Channel
         if and(min(Fib.PosYAll(ii,i))==0,min(Fib.PosZAll(ii,i))==0);
            Fib.veloX(ii,i)=Umax;
         end

         % Elimiate negative Values
          if Fib.veloX(ii,i) < 0;
             Fib.veloX(ii,i) =0;
          end
      end
      plot3(Fib.PosYAll(:,1),Fib.PosZAll(:,1),Fib.veloX(:,1),'.');
      hold on;
    end
end
hold off;

%% ---------------------- Jeffery Orbit ------------------------------ %
Fib=func_JefferyOrbit(Fib);

%% ------------------- Write Output File ----------------------------- %
%% Create txt - File
% create_atoms    1 single initialPosX initialPosY1 initialPosZ1 units box
    ID =1;
for Type=1:1:Fib.nAtomType
 % Open output file
 InitFibPos=fopen(...
  fullfile(resDir,['in.ParticlePos_AR',num2str(Fib.AR(Type)),'.txt']),'w');
    for i=1:1:size(Fib.PosXAll,1);
       if or(isnan(Fib.PosYAll(i,Type)),isnan(Fib.PosZAll(i,Type)))
       else
         if Fib.setVeloX == 1 % Output with set velocities
         fprintf(InitFibPos,...
               '%s %i %s %5.4f %5.4f %5.4f %s \n %s %i %s %5.4f \n',...
               'create_atoms    ',Type+(Fib.startType-1),... % create atom
               'single ', Fib.PosXAll(i,Type),...
                Fib.PosYAll(i,Type),...
                Fib.PosZAll(i,Type),...
               ' units box ',...
               ' set atom ',...  % set atom ID vx veloX
               ID,...
               ' vx ',...
               Fib.veloX(i,Type));
           ID =ID+1;

           else % Output without set velocities
                fprintf(InitFibPos, '%s %i %s %5.4f %5.4f %5.4f %s \n',...
               'create_atoms    ',Type+(Fib.startType-1),... % create atom
               'single ', Fib.PosXAll(i,Type),...
                Fib.PosYAll(i,Type),...
                Fib.PosZAll(i,Type),...
               ' units box ');
           end
       end
    end
end
disp('.txt-File is done!')

%% ------------------- Plot Fibers at initial position ----------------- %
funcStat=0;
```

```
funcStat=func_PlotInitialPos(Fib.nAtomType,Fib,Box);

%% ---------------------------- End of Program -------------------------- %
disp('End of program - Have fun...')
```

```
function funcStat=func_PlotInitialPos(nAtomType,Fib,Box);

% Plot initial positions of all fibers in channel with square cross section

global interpreterName resDir figVisible

%% ------------------- Formating ---------------------------------------%
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

%% -------------------  Figure / Plot  ---------------------------------%
% ------------------- Plot yz-Plane -------------------------------%
figName = 'Initial position of all fibers yz';
scrs = get(0,'screensize');
rect = [10, 10, 600, 500]; %[left, bottom, width, height]

fig1 = figure('Name',figName,'Position',rect,'Visible',figVisible);

% subplot(2,1,1) % yz
for Type=1:1:nAtomType
        plot(Fib.PosYAll(:,Type), Fib.PosZAll(:,Type),...
         symbolArray{Type},...
        'MarkerSize',markersize*Type*0.1);hold on;
    legendText{Type}=...
        [preAfterSymbol,'AR=',num2str(Fib.AR(Type)),preAfterSymbol];
   end

 legend(legendText,'Location','Northwest');

 if(  strcmp(interpreterName, 'latex') )
   legend(legendText,'fontsize',fontSizeLabel*0.85,...
      'interpreter',interpreterName,'Location','Northwest');
 end

   % Lines
   % plot([Box.yMin Box.yMax],[0 0],'k-.','MarkerSize',2);hold on;
   % plot([0 0],[Box.zMin Box.zMax],'k-.','MarkerSize',2);hold off;

   axis([Box.yMin Box.yMax Box.zMin Box.zMax]);
   axis square;
   set(gca,'ZLim',[Box.zMin Box.zMax]);
   set(gca,'YLim',[Box.yMin Box.yMax]);
%     title([preAfterSymbol,'initial fiber distribution',' y^{+}-z^{+}',...
%          preAfterSymbol],...
%         'Interpreter',interpreterName,'fontsize',fontSizeTitle);
   xlabel([preAfterSymbol,'y^{+}',preAfterSymbol],...
       'Interpreter',interpreterName);
   ylabel([preAfterSymbol,'z^{+}',preAfterSymbol],...
       'Interpreter',interpreterName);

   grid off
   box on
   legend boxoff
   set(0,'defaultaxesfontsize',defaultaxesfontsize);
   set(0,'defaulttextfontsize',defaulttextfontsize);
   set(gca,'FontSize',fontSizeAxis);
   set(gca,'FontWeight','normal');
   xlhand = get(gca,'xlabel');
   ylhand = get(gca,'ylabel');
   zlhand = get(gca,'zlabel');
   set(xlhand,'fontsize',fontSizeLabel);
   set(ylhand,'fontsize',fontSizeLabel);
   set(zlhand,'fontsize',fontSizeLabel)
   set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
   set(xlhand,'FontWeight','bold');
   set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
   set(ylhand,'FontWeight','bold');
   set(zlhand,'FontName','Times'); set(zlhand,'FontAngle','italic');
```

```matlab
% Jeffery Orbit Calculation and plot
function Fib=func_JefferyOrbit(Fib)

global interpreterName resDir currDir figVisible halfLength Umax

%% ------------------- Formating ---------------------------------------%
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

%% ----------------- Jeffery Orbit Calculations -----------------------%
for j=1:1:Fib.nAtomType
    for i=1:1:Fib.nFibZ

    % Velocity
    Fib.Class(j).veloX(i,1)=velSCS(halfLength, Umax,...
      Fib.Class(j).r(i,1), Fib.Class(j).theta(i,1));

    % Shear Rate
    Fib.Class(j).ShearRate(i,1) = ShearRateSCS(halfLength, Umax,...
      Fib.Class(j).r(i,1), Fib.Class(j).theta(i,1));

    % Orbit Period
    Fib.Class(j).tOrbit(i,1)=...
     2*pi.*(Fib.AR(j)+1./Fib.AR(j))./Fib.Class(j).ShearRate(i,1);

    % Half Orbit (pi = 180)
     Fib.Class(j).tOrbitHalf(i,1)=Fib.Class(j).tOrbit(i,1)*0.5;

    % Length for Orbit
     Fib.Class(j).sOrbitHalf(i,1)=...
      Fib.Class(j).veloX(i,1)*Fib.Class(j).tOrbitHalf(i,1);
     end

end

%% -------------------------- Figure / Plot  ---------------------------%
% ----------------- Length of Traveling for 0.5 orbit -----------------
figName = 's0.5Obrit_zInit';
scrs = get(0,'screensize');
rect = [10, 10, 600, 500]; %[left, bottom, width, height]
fig1 = figure('Name',figName,'Position',rect,'Visible',figVisible);
legendText={};

for j=1:1:Fib.nAtomType
    x=Fib.Class(j).zPos(:,1);
    y=Fib.Class(j).sOrbitHalf(:,1);

    plot(x,y,symbolArray{j});hold on;

    legendText{j}=...
        [preAfterSymbol,'AR=',num2str(Fib.AR(j)),preAfterSymbol];
end

legend(legendText,'Location','Northwest');

 if(  strcmp(interpreterName, 'latex') )
   legend(legendText,'fontsize',fontSizeLabel*0.85,...
       'interpreter',interpreterName,'Location','Northwest');
 end

xlabel([preAfterSymbol,'z^{+}',preAfterSymbol],...
    'interpreter',interpreterName)
ylabel([preAfterSymbol,'s^{+}_{0.5 orbit}',preAfterSymbol],...
    'interpreter',interpreterName)

    axis square
    grid off
    box on
    set(zlhand,'FontWeight','bold');

% ------------------- Save Figure ----------
Name='Particle_InitialPosition_yz';
printAs='-dpdf';
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 23])
print(printAs,'-r450', [resDir,Name])

% ------------------- Plot xy-Plane --------------------------------%
figName = 'Initial position of all fibers xy';
scrs = get(0,'screensize');
rect = [10, 10, 900, 600]; %[left, bottom, width, height]

fig1 = figure('Name',figName,'Position',rect,'Visible',figVisible);
% subplot(2,1,2) % xy
    for Type=nAtomType:-1:1
        plot(Fib.PosXAll(:,Type), Fib.PosYAll(:,Type),...
          symbolArray{Type},...
         'MarkerSize',markersize*0.25);hold on;
    end
    axis([Box.xMin Box.xMax Box.zMin Box.zMax])
    axis equal
    set(gca,'xLim',[Box.xMin Box.xMax])
    set(gca,'YLim',[Box.zMin Box.zMax])
%     title([preAfterSymbol,'initial fiber distribution','x^{+}-y^{+}',...
%          preAfterSymbol],...
%         'Interpreter',interpreterName,'fontsize',fontSizeTitle);
    xlabel([preAfterSymbol,'x^{+}',preAfterSymbol],...
        'Interpreter',interpreterName);
     ylabel([preAfterSymbol,'y^{+}',preAfterSymbol],...
        'Interpreter',interpreterName);

     grid off
     box on
     legend boxoff
     set(0,'defaultaxesfontsize',defaultaxesfontsize);
     set(0,'defaulttextfontsize',defaulttextfontsize);
     set(gca,'FontSize',fontSizeAxis);
     set(gca,'FontWeight','normal');
     xlhand = get(gca,'xlabel');
     ylhand = get(gca,'ylabel');
     zlhand = get(gca,'zlabel');
     set(xlhand,'fontsize',fontSizeLabel);
     set(ylhand,'fontsize',fontSizeLabel);
     set(zlhand,'fontsize',fontSizeLabel)
     set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
     set(xlhand,'FontWeight','bold');
     set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
     set(ylhand,'FontWeight','bold');
     set(zlhand,'FontName','Times'); set(zlhand,'FontAngle','italic');
     set(zlhand,'FontWeight','bold');

% ------------------- Save Figure ----------
Name='Particle_InitialPosition_xy';
printAs='-dpdf';
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 23])
print(printAs,'-r450', [resDir,Name])

funcStat=1;
end
```

```
   legend boxoff
   set(0,'defaultaxesfontsize',defaultaxesfontsize);
   set(0,'defaulttextfontsize',defaulttextfontsize);
   set(gca,'FontSize',fontSizeAxis);
   set(gca,'FontWeight','normal');
   xlhand = get(gca,'xlabel');
   ylhand = get(gca,'ylabel');
   zlhand = get(gca,'zlabel');
   set(xlhand,'fontsize',fontSizeLabel);
   set(ylhand,'fontsize',fontSizeLabel);
   set(zlhand,'fontsize',fontSizeLabel)
   set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
   set(xlhand,'FontWeight','bold');
   set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
   set(ylhand,'FontWeight','bold');
   set(zlhand,'FontName','Times'); set(zlhand,'FontAngle','italic');
   set(zlhand,'FontWeight','bold');

% Save Figure
Name='Particle_InitialPosition_s0.5orbit';
printAs='-dpdf';
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 23])
print(printAs,'-r450', [resDir,Name])

% -------------------------- Time for 0.4 orbit -------------------------
figName = 't0.5Obrit_zInit';
scrs = get(0,'screensize');
rect = [10, 10, 600, 500]; %[left, bottom, width, height]

fig2 = figure('Name',figName,'Position',rect,'Visible',figVisible);

legendText={};

for j=1:1:Fib.nAtomType
    x=Fib.Class(j).zPos(:,1);
    y=Fib.Class(j).tOrbitHalf(:,1);

    plot(x,y,symbolArray{j});hold on;

    legendText{j}=...
        [preAfterSymbol,'AR=',num2str(Fib.AR(j)),preAfterSymbol];
end

legend(legendText,'Location','Northwest');

 if(  strcmp(interpreterName, 'latex') )
   legend(legendText,'fontsize',fontSizeLabel*0.85,...
       'interpreter',interpreterName,'Location','Northwest');
 end


xlabel([preAfterSymbol,'z^{+}',preAfterSymbol],...
    'interpreter',interpreterName)
ylabel([preAfterSymbol,'t^{+}_{0.5 orbit}',preAfterSymbol],...
    'interpreter',interpreterName)

    axis square
    grid off
    box on
    legend boxoff
    set(0,'defaultaxesfontsize',defaultaxesfontsize);
    set(0,'defaulttextfontsize',defaulttextfontsize);
    set(gca,'FontSize',fontSizeAxis);
    set(gca,'FontWeight','normal');
    xlhand = get(gca,'xlabel');
    ylhand = get(gca,'ylabel');
    zlhand = get(gca,'zlabel');
    set(xlhand,'fontsize',fontSizeLabel);
    set(ylhand,'fontsize',fontSizeLabel);
    set(zlhand,'fontsize',fontSizeLabel)
    set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
    set(xlhand,'FontWeight','bold');
    set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
    set(ylhand,'FontWeight','bold');
    set(zlhand,'FontName','Times'); set(zlhand,'FontAngle','italic');
    set(zlhand,'FontWeight','bold');

%% -------------------- Save Figure --------------------------------%
Name='Particle_InitialPosition_t0.5orbit';
printAs='-dpdf';
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 23])
print(printAs,'-r450', [resDir,Name])



end

```

```
% Program to ProstProcess CFDEM Data generated
% with CFDEM(R) simulation:
% Critical Line for separation
% Orientation before and after Junction
% (c) Lisa Koenig, TU Graz,
%
% Required outMass File:
% ID diameter x y z u v w  (ex ey ez, color)
%  1     2    3 4 5 6 7 8    9 10 11
%
% Required dump:
% 1           ... id
% 2           ... type
% 3 4 5       ... x y z
% 6 7 8       ... vx vy vz
% 9 10 11     ... fx fy fz
% 12 13 14    ...omegax omegay omegaz
% 15          ... radius
% 16 17 18    ... f_ex[1] f_ex[2] f_ex[3]
% 19          ... f_shape[1]
% 20 21 22 23 ...c_cQuatw c_cQuati c_cQuatj c_cQuatk

clear all; close all; clc;

% Define Global Values
global  homeDir inDirOutMass inDirDump resDir ...          % global directories
        FigVisible printAs printRes ...   % global plot specifications
        interpreterName %caseIdToRun      % global external data

%% --------------------------- User Input ----------------------------- %
% Directories
homeDir=pwd;
inDirOutMass='../../outMass/';
inDirDump='../../post/';
resDir='../../postProcessing/';

preTFile='outMassPreT.dat.1';
postTFile='outMassPostT.dat.1';

outFileName='CriticalLine_SCW0.5_45_Re_500_dp0.10_xDir.txt';
Re=500;
dp=0.10;        % dp Case
SCW=0.5;     % side Channel Width
angle=45;
halfLength=0.5; % Height of Channel

% Plot specifications
FigVisible='off'; % 'off' / 'on' show figures
printAs='dpng';   % 'dpng' 'dpdf' ... format plots are safed
printRes=400;     %  plot resultion

% Plane in Main Channel specifying separation
Plane.MainX=0.7;

% Plane in Side Channel specifying separation
Plane.SideZ=-0.51;

%% -----------------------  Environment Setting ----------------------- %
% Check if octave of matlab is used
tempFile = fopen('tempDat.txt');
C = textscan(tempFile,'%d',1);
fclose(tempFile);
interPreterName=C{1,1};
%caseIdToRun=C{1,2};

if interPreterName == 1 % MATLAB
    interpreterName='latex';
end

if interPreterName == 2 % OCTAVE
    interpreterName='tex';
end

% Check Folders, if they don't exist create it
if exist(resDir) == 0 % if folder does not exist create it!
    mkdir(resDir);
    disp([['Folder: ', fullfile(resDir),' created']]);
end

%% --------------------------- Read Data ------------------------------ %

%   outMass Data
% ----------------
Data.preT  = dlmread(fullfile(inDirOutMass,preTFile),'',1,0);
Data.preT  = sortrows(Data.preT,1); % sort acc. ID

Data.postT = dlmread(fullfile(inDirOutMass,postTFile),'',1,0)
Data.postT = sortrows(Data.postT,1); % sort acc. ID

%    Last Dump
% --------------
% Check number of dump-files in folder
cd(inDirDump);
files=dir('dump*.liggghts'); % list all dump-files
cd(homeDir);
nDmpFiles=length(files);

% List all name-numbers of the dump-files and find first and last
for i=1:1:nDmpFiles
files(i).numbers=sscanf(files(i).name,'dump%i%.liggghts');
end

tempFileNumbers=([files.numbers]);
tempFileNumbers=sort(tempFileNumbers,2);

FstDmp=tempFileNumbers(1)   % first  Dump File number (e.g. start of sim)
LstDmp=tempFileNumbers(end) % last  Dump File number (e.g. end of sim)

% Read first Dump
dmpFst= ...
        readdump_all(...
        fullfile(inDirDump,['dump',num2str(FstDmp,'%i'),'.liggghts']));
dmpFst.atom_data=sortrows(dmpFst.atom_data,1); % sort acc. ID

% Read last Dump
dmpLst= ...
        readdump_all(...
        fullfile(inDirDump,['dump',num2str(LstDmp,'%i'),'.liggghts']));
dmpLst.atom_data=sortrows(dmpLst.atom_data,1); % sort acc. ID

% Number of Fibres initiated
Fib.nTotalFst=length(dmpFst.atom_data(:,1));
Fib.nTotalLst=length(dmpLst.atom_data(:,1));

% Atom Types and dMajor
Fib.AtomTypes=unique(dmpFst.atom_data(:,2));
Fib.nAtomTypes=length(Fib.AtomTypes);
Fib.dMajor=unique(dmpLst.atom_data(:,19))*2;

for j=1:1:Fib.nAtomTypes
    Fib.Class(j).type=Fib.AtomTypes(j);
end

%% ------------------ Check if all particle are gone ------------------ %
% gone: postT or in side channel
count=1;
Fib.NotYet=[];
for i=1:1:Fib.nTotalLst
    tempX=dmpLst.atom_data(i,3); % particle x-position
    tempZ=dmpLst.atom_data(i,5); % particle y-position

    if and(tempX < Plane.MainX, tempZ > Plane.SideZ)
       Fib.NotYet(count,:)=dmpLst.atom_data(i,:);
       count=count+1;
    end
end

if size(Fib.NotYet,1)~= 0
    disp('WARNING NOT ALL FIBRES ARE ASSIGNED!')
    Fib.NotYet
end

%% --------------------- Critical Separation z+ ------------------------- %

ID.preT=Data.preT(:,1);          % ID of all fibres pre-side-channel
ID.postT=Data.postT(:,1);        % ID of all fibres post-side-channel
ID.Sep=setdiff(ID.preT,ID.postT);% ID of all fibres separated

count=1;
for i=1:1:length(ID.Sep)
id=ID.Sep(i);
Fib.Sep(count,:)=dmpFst.atom_data(id,:);
count=count+1;
end

% Divide Separated Fibres in Classes acc. to Type
for j=1:1:Fib.nAtomTypes
    count=1;
    for i=1:1:size(Fib.Sep,1)
       if Fib.Sep(i,2) == Fib.AtomTypes(j)
          Fib.Class(j).Sep(count,:)=Fib.Sep(i,:);
          count=count+1;
       end
    end
end

% Get highest z+ - value for separated Fibre in each Class
for j=1:1:Fib.nAtomTypes
    Fib.Class(j).maxZ=max(Fib.Class(j).Sep(:,5)); % z-value
    Fib.Class(j).DistToWall=halfLength-abs(Fib.Class(j).maxZ);
    Fib.Class(j).DmajorHalf=Fib.dMajor(j)/2;
    Fib.Class(j).DistToWallToDmajorHalf=...
        Fib.Class(j).DistToWall/Fib.dMajor(j)/2;
end

%% ------------------------- Write output file -------------------------- %
outFile=fopen(fullfile(resDir,outFileName),'w');

fprintf(outFile,'%s \n',...
 '# Re  dp  SCW  angle Type   z+_max_sep   distToWall+  dMajor  dMajor/2');
for j=1:1:Fib.nAtomTypes
fprintf(outFile,'%4.2f %3.2f %3.2f %3.1f %i %8.6f %8.6f %5.3f %5.3f\n',...
          Re,...
          dp,...
          SCW,...
          angle,...
          Fib.Class(j).type,...
          Fib.Class(j).maxZ,...
          halfLength-abs(Fib.Class(j).maxZ),...
          Fib.dMajor(j),...
          Fib.dMajor(j)/2);
end

close all
disp('End of program. Have fun...')

```

```
% Program to ProstProcess CFDEM Data generated
% with CFDEM(R) simulation:
% Critical Line for separation
% Orientation before and after Junction
% (c) Lisa Koenig, TU Graz,
%
% Required outMass File:
% ID diameter x y z u v w  (ex ey ez, color)
%  1     2    3 4 5 6 7 8    9 10 11
%
% Required dump:
% 1           ... id
% 2           ... type
% 3 4 5       ... x y z
% 6 7 8       ... vx vy vz
% 9 10 11     ... fx fy fz
% 12 13 14    ...omegax omegay omegaz
% 15          ... radius
% 16 17 18    ... f_ex[1] f_ex[2] f_ex[3]
% 19          ... f_shape[1]
% 20 21 22 23 ...c_cQuatw c_cQuati c_cQuatj c_cQuatk

clear all; close all; clc;

% Define Global Values
global  homeDir inDirOutMass inDirDump resDir ...          % global directories
        FigVisible printAs printRes ...   % global plot specifications
        interpreterName %caseIdToRun      % global external data

%% ---------------------------- User Input ------------------------------ %
% Directories
homeDir=pwd;
inDirOutMass='../../outMass/';
inDirDump='../../post/';
resDir='../../postProcessing/';

preTFile='outMassPreT.dat.1';
postTFile='outMassPostT.dat.1';

outFileName='CriticalLine_SCW1_90_Re_500_dp0.35_xDir.txt';
Re=500;
dp=0.35;        % dp Case
SCW=1;      % side Channel Width
angle=90;
halfLength=0.5; % Height of Channel
Fib.dMinor=0.02;

% Plot specifications
FigVisible='on'; % 'off' / 'on' show figures
printAs='-dpng';   % 'dpng' 'dpdf' ... format plots are safed
printRes=400;     %  plot resultion

% Plane in Main Channel specifying separation
Plane.MainX=0.7;

% Plane in Side Channel specifying separation
Plane.SideZ=-0.51;

%% -----------------------  Environment Setting ----------------------- %
% Check if octave of matlab is used
tempFile = fopen('tempDat.txt');
C = textscan(tempFile,'%d',1);
fclose(tempFile);
interPreterName=C{1,1};
%caseIdToRun=C{1,2};

if interPreterName == 1 % MATLAB
    interpreterName='latex';
end

```

```
if interPreterName == 2 % OCTAVE
    interpreterName='tex';
end

% Check Folders, if they don't exist create it
if exist(resDir) == 0 % if folder does not exist create it!
    mkdir(resDir);
    disp({['Folder: ', fullfile(resDir),' created']});
end

%% ---------------------------- Read Data ------------------------------ %

%   outMass Data
% ----------------
Data.preT  = dlmread(fullfile(inDirOutMass,preTFile),'',1,0);
Data.preT  = sortrows(Data.preT,1); % sort acc. ID

Data.postT = dlmread(fullfile(inDirOutMass,postTFile),'',1,0)
Data.postT = sortrows(Data.postT,1); % sort acc. ID

%    Last Dump
% --------------
% Check number of dump-files in folder
cd(inDirDump);
files=dir('dump*.liggghts'); % list all dump-files
cd(homeDir);
nDmpFiles=length(files);

% List all name-numbers of the dump-files and find first and last
for i=1:1:nDmpFiles
files(i).numbers=sscanf(files(i).name,'dump%i%.liggghts');
end

tempFileNumbers=([files.numbers]);
tempFileNumbers=sort(tempFileNumbers,2);

FstDmp=tempFileNumbers(1)   % first  Dump File number (e.g. start of sim)
LstDmp=tempFileNumbers(end) % last  Dump File number (e.g. end of sim)

% Read first Dump
dmpFst= ...
        readdump_all(...
        fullfile(inDirDump,['dump',num2str(FstDmp,'%i'),'.liggghts']));
dmpFst.atom_data=sortrows(dmpFst.atom_data,1); % sort acc. ID

% Read last Dump
dmpLst= ...
        readdump_all(...
        fullfile(inDirDump,['dump',num2str(LstDmp,'%i'),'.liggghts']));
dmpLst.atom_data=sortrows(dmpLst.atom_data,1); % sort acc. ID

% Number of Fibres initiated
Fib.nTotalFst=length(dmpFst.atom_data(:,1));
Fib.nTotalLst=length(dmpLst.atom_data(:,1));

% Atom Types and dMajor
Fib.AtomTypes=unique(dmpFst.atom_data(:,2));
Fib.nAtomTypes=length(Fib.AtomTypes);
Fib.dMajor=unique(dmpLst.atom_data(:,19))*2;
Fib.AR=Fib.dMajor./Fib.dMinor;
Fib.dSphere=unique(Data.preT(:,2));  % sphere aquiv. diameter
% Data.dMajor=round(Data.dSphere.^3./Data.dMinor^2*100)/100; % dMajor axis

for j=1:1:Fib.nAtomTypes
    Fib.Class(j).type=Fib.AtomTypes(j);
end

%% ------------------- Check if all particle are gone ------------------- %
% gone: postT or in side channel
count=1;
Fib.NotYet=[];
for i=1:1:Fib.nTotalLst
    tempX=dmpLst.atom_data(i,3); % particle x-position
    tempZ=dmpLst.atom_data(i,5); % particle y-position

    if and(tempX < Plane.MainX, tempZ > Plane.SideZ)
       Fib.NotYet(count,:)=dmpLst.atom_data(i,:);
       count=count+1;
    end
end

if size(Fib.NotYet,1)~= 0
    disp('WARNING NOT ALL FIBRES ARE ASSIGNED!')
    Fib.NotYet
    size(Fib.NotYet,1)
end

%% -------------------- Critical Separation z+ ------------------------ %

ID.preT=Data.preT(:,1);          % ID of all fibres pre-side-channel
ID.postT=Data.postT(:,1);        % ID of all fibres post-side-channel
ID.Sep=setdiff(ID.preT,ID.postT);% ID of all fibres separated

count=1;
for i=1:1:length(ID.Sep)
id=ID.Sep(i);
Fib.Sep(count,:)=dmpFst.atom_data(id,:);
count=count+1;
end

% Divide All Fibres in Classes
for j=1:1:Fib.nAtomTypes
    count=1;
    for i=1:1:size(Data.preT,1)
       if Data.preT(i,2) == Fib.dSphere(j)
          Fib.Class(j).All(count,:)=Data.preT(i,:);
          count=count+1;
       end
    end
end

% Divide Not-Separated Fibres in Classes in postT
for j=1:1:Fib.nAtomTypes
    count=1;
    for i=1:1:size(Data.postT,1)
       if Data.postT(i,2) == Fib.dSphere(j)
          Fib.Class(j).NotSep(count,:)=Data.postT(i,:);
          count=count+1;
       end
    end
end


% Divide Separated Fibres in Classes acc. to Type
for j=1:1:Fib.nAtomTypes
    count=1;
    Fib.Class(j).Sep(count,:)=ones(1,size(Fib.Sep(1,:),2))*NaN;
    for i=1:1:size(Fib.Sep,1)
       if Fib.Sep(i,2) == Fib.AtomTypes(j)
          Fib.Class(j).Sep(count,:)=Fib.Sep(i,:);
          count=count+1;
       end
    end
end

% Get highest z+ - value for separated Fibre in each Class
for j=1:1:Fib.nAtomTypes
    Fib.Class(j).maxZ=max(Fib.Class(j).Sep(:,5)); % z-value
    Fib.Class(j).DistToWall=halfLength-abs(Fib.Class(j).maxZ);
    Fib.Class(j).DmajorHalf=Fib.dMajor(j)/2;
```

```
   Fib.Class(j).DistToWallToDmajorHalf=...
      Fib.Class(j).DistToWall/Fib.dMajor(j)/2;
end

% Determine downwards movement of all firbes that are noit separated
for j=1:1:Fib.nAtomTypes
   for i=1:1:size(Fib.Class(j).NotSep,1)

      zPostT=Fib.Class(j).NotSep(i,5); % z-position after T

      Index=find(Data.preT(:,1)==Fib.Class(j).NotSep(i,1));
      zPreT=Data.preT(Index,5);        % z-position before T

      dz=zPreT-zPostT;

      Fib.Class(j).NotSepDz(i,1)=Fib.Class(j).NotSep(i,1); % ID
      Fib.Class(j).NotSepDz(i,2)=zPreT;
      Fib.Class(j).NotSepDz(i,3)=zPostT;
      Fib.Class(j).NotSepDz(i,4)=halfLength+zPreT;
      Fib.Class(j).NotSepDz(i,5)=halfLength+zPostT;
   end

   % Sort row by dz
   Fib.Class(j).NotSepDz=sortrows(Fib.Class(j).NotSepDz,4);
end

%% Plot Downwards movement deltaZ
% Get Screen Size
%scrsz = get(groot,'ScreenSize');% MATLAB >R2014b
scrsz = get(0,'ScreenSize'); % MATLAB <R2014b
rect = [10, 10, 600, 600]; %[left, bottom, width, height]
Name='DownMoveT_xDir';

% Formating
run('formatting_MATLAB');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

FigSepdZ=figure('Name',Name,'Position',...
   rect,'Visible',FigVisible);

for j=1:1:Fib.nAtomTypes

  plot(Fib.Class(j).NotSepDz(:,4),...
      (Fib.Class(j).NotSepDz(:,3)-Fib.Class(j).NotSepDz(:,2))./...
       Fib.Class(j).NotSepDz(:,4),...
      symbolArrayC{j},...
      'markersize',markersize); hold on;

   legendtext{j}=...
   [preAfterSymbol,'AR=',num2str(Fib.AR(j)),preAfterSymbol];

end



 if(  strcmp(interpreterName, 'tex') )
   legend(legendtext,'Location','northeast',...
      'Orientation','vertical');
   xlabel([preAfterSymbol,'s+z^{+}_{preT}',preAfterSymbol],...
         'FontSize',fontSizeLabel);
   ylabel([preAfterSymbol,...
      '(z^{+}_{preT}-z^{+}_{postT})/(s+z^{+}_{preT})',...
         preAfterSymbol],...
         'FontSize',fontSizeLabel);
 end

 if(  strcmp(interpreterName, 'latex') )
  legend(legendtext,'Location','NorthEast',...
  'interpreter',interpreterName,...
  'FontSize',fontSizeLabel,'Orientation','vertical');

  xlabel([preAfterSymbol,'s+z^{+}_{preT}',preAfterSymbol],...
        'interpreter',interpreterName,'FontSize',fontSizeLabel);

  ylabel([preAfterSymbol,...
     '(z^{+}_{preT}-z^{+}_{postT})/(s+z^{+}_{preT})',...
        preAfterSymbol],...
        'interpreter',interpreterName,'FontSize',fontSizeLabel);
 end

% axis([0 22 0 15])

grid off;
box on;
legend boxoff;
% axis square;
% set(gca,'XTick',[2 5 10 15 20]);
% set(gca,'YTick',[-0.5:0.25:0.5]);
set(gca,'XMinorTick','on','YMinorTick','on')
set(0,'defaultaxesfontsize',stdTextFontSize);
set(0,'defaulttextfontsize',stdTextFontSize);
set(gca,'FontSize',stdTextFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',fontSizeLabel); set(ylhand,'fontsize',fontSizeLabel);
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters');

% Save Figure
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 18]);
print(printAs,'-r300', [resDir,Name]);



%% ------------------------- Write output file -------------------------- %
outFile=fopen(fullfile(resDir,outFileName),'w');

fprintf(outFile,'%s \n',...
 '# Re  dp  SCW  angle Type   z+_max_sep   distToWall+  dMajor  dMajor/2');
for j=1:1:Fib.nAtomTypes
fprintf(outFile,'%4.2f %3.2f %3.2f %3.1f %i %8.6f %8.6f %5.3f %5.3f\n',...
        Re,...
        dp,...
        SCW,...
        angle,...
        Fib.Class(j).type,...
        Fib.Class(j).maxZ,...
        halfLength-abs(Fib.Class(j).maxZ),...
        Fib.dMajor(j),...
        Fib.dMajor(j)/2);
end


disp('End of program. Have fun...')

```

```
%% This preprocessing program provides Fibre Data (position and velocity)
% (e.g for usage in a CFDEM simulation)
% Lisa Koenig, August 2015, TUGraz
%
%  Schematic Diagramm:
% --------------------
%
%          Z-dir.                Z-dir.
%            /\                   /\
%            ||                   ||
%            ||                   ||
%         +--------+    +--------------------------------+
%         |      |   |                    |
%  Y-dir.<- |      |   |                    | -> X-dir.
%         |      |   |                    |
%         +--------+    +-----------+---------+----------+
%                          |      |
%                          |      |
%
% The Fibre initial velocity can be set with the Velocity profil given by:
% Tamayol, Bahrami - Laminar Flow in Mircochannels With
% Noncircular Cross Section
% setVeloX=0 ... initial fibre velocity is zero
% setVeloX=1 ... initial fibre velocity is calculated

clear all; clc; close all; fclose all
global interpreterName resDir currDir figVisible halfLength Umax

%% -------------------- Directories ------------------------------------ %
inDir = '../../input/';
fileName= 'input.txt';
resDir='../../preProcessing/';
currDir=pwd;

figVisible='on';

%% -------------------- Read temporary file for setting ----------------- %
% Check if octave of matlab is used
tempDataFile = fopen('tempData.txt');
C = textscan(tempDataFile,'%f %f',1);
fclose(tempDataFile);
InterPreterName=C{1,1};

if InterPreterName == 1 % MATLAB
    interpreterName='latex'
end

if InterPreterName == 2 %OCTAVE
    interpreterName='tex'
end

%% -------------------- Prepare environment ----------------------------- %
% Check if output directory exists
cd ..
cd ..
if exist('preProcessing') == 0 % if folder does not exist create it!
   mkdir('preProcessing');
end
cd (currDir)

%% -------------------- Read Input Data --------------------------------- %

    % Get Data from input File via dlmread()
    temp.Data1 = dlmread(strcat(inDir,fileName),'',[15 0 15 5]);
    temp.Data2 = dlmread(strcat(inDir,fileName),'',[18 0 18 10]);
    temp.Data3 = dlmread(strcat(inDir,fileName),'',21,0);

    % Box Dimensions
    Box.xMin     = temp.Data1(1);
    Box.xMax     = temp.Data1(2);
    Box.yMin     = temp.Data1(3);
    Box.yMax     = temp.Data1(4);
    Box.zMin     = temp.Data1(5);
    Box.zMax     = temp.Data1(6);

    % Fibre/Fluid Data
    Fib.rho  = temp.Data2(1);
    rhoFluid = temp.Data2(2);

    % Set velocity
    Fib.nFibZ    = temp.Data2(3);
    Fib.setVeloX = temp.Data2(4);
    Umax         = temp.Data2(5);
    halfLength   = temp.Data2(6);
    Fib.x        = temp.Data2(7);
    Fib.y        = temp.Data2(8);
    Fib.zmin     = temp.Data2(9);
    Fib.zmax     = temp.Data2(10);
    Fib.startType= temp.Data2(11);

    % Fibre Data
    Fib.Data = sortrows(temp.Data3,[1 2]); % Sort acc. to dMajor
    Fib.dMajor = Fib.Data(:,1);
    Fib.dMinor = Fib.Data(:,2);

    clear temp.*; % clear temporary data


%% -------------------- Fibre Values Calculation ------------------------ %
% Fibre Classes
Fib.nAtomType = length(Fib.dMajor);
% Diameter of Spherocylinder
Fib.dCylinder =1.24.*Fib.dMinor./sqrt(log(Fib.dMajor./Fib.dMinor));
% Aspect Ratios
Fib.AR        =Fib.dMajor./Fib.dMinor;
% Vol. single fibre in class i; fibre assumed to be ellipsoid
Vol.Fibi      = Fib.dMajor.^3 ./ Fib.AR .^2 .* pi ./ 6;
% mass of one single fibre in class i
mass.Fibi     = Fib.rho * Vol.Fibi;
% vol. equiv. diameter
Fib.dSphere   = (Fib.dMinor.*Fib.dMinor.*Fib.dMajor).^(1/3);

%% -------------------- Fibre Positioning ----------------------------- %

% Distance between Fibres
zDist=-(Fib.zmax-Fib.zmin)/(Fib.nFibZ-1);
Fib.PosXAll=[]; Fib.PosYAll=[];Fib.PosZAll=[];

for j=1:1:Fib.nAtomType
    for i=1:1:Fib.nFibZ
        Fib.Class(j).xPos(i,1)=Fib.x;       % x-postion
        Fib.Class(j).yPos(i,1)=Fib.y;       % y-postion
        Fib.z=Fib.zmax+(i-1)*zDist;         % z-postion
        Fib.Class(j).zPos(i,1)=Fib.z;

        Fib.Class(j).r(i,1)=(Fib.z^2+Fib.y^2)^0.5;
        Fib.Class(j).theta(i,1)=atan(Fib.y/Fib.z);
        if Fib.z==0
            Fib.Class(j).theta(i,1)=0;
        end
    end

    % All Position
    Fib.PosXAll=horzcat(Fib.PosXAll, Fib.Class(j).xPos(:,1));
    Fib.PosYAll=horzcat(Fib.PosYAll, Fib.Class(j).yPos(:,1));
    Fib.PosZAll=horzcat(Fib.PosZAll, Fib.Class(j).zPos(:,1));

end

%% ------------------- Fibre Velocity ---------------------------------- %
```

```
% set Inlet Velocity in x direction acc. to Tamayol, Bahrami - Laminar Flow
% in Mircochannels With Noncircular Cross Section
% Set initial velocity

if Fib.setVeloX == 1
   for i=1:1:Fib.nAtomType
   r = sqrt(Fib.PosYAll(:,i).^2 + Fib.PosZAll(:,i).^2);
   theta = acos(Fib.PosYAll(:,i)./r);
   tempVeloX = velSCS(halfLength, Umax, r, theta);
   Fib.veloX(:,i)=tempVeloX;
     for ii=1:1:size(Fib.veloX(:,i),1);
       % Elimiate NAN velue at center of Channel
        if and(min(Fib.PosYAll(ii,i))==0,min(Fib.PosZAll(ii,i))==0);
           Fib.veloX(ii,i)=Umax;
        end

        % Elimiate negative Values
         if Fib.veloX(ii,i) < 0;
            Fib.veloX(ii,i) =0;
         end
     end
     plot3(Fib.PosYAll(:,1),Fib.PosZAll(:,1),Fib.veloX(:,1),'.');
     hold on;
   end
end
hold off;

%% ---------------------- Jeffery Orbit ------------------------------- %
Fib=func_JefferyOrbit(Fib);

%% ------------------- Write Output File ----------------------------- %
%% Create txt - File
% create_atoms    1 single initialPosX initialPosY1 initialPosZ1 units box
   ID =1;
for Type=1:1:Fib.nAtomType
 % Open output file
 InitFibPos=fopen(...
  fullfile(resDir,['in.ParticlePos_AR',num2str(Fib.AR(Type)),'.txt']),'w');
   for i=1:1:size(Fib.PosXAll,1);
      if or(isnan(Fib.PosYAll(i,Type)),isnan(Fib.PosZAll(i,Type)))
      else
        if Fib.setVeloX == 1 % Output with set velocities
        fprintf(InitFibPos,...
              '%s %i %s %5.4f %5.4f %5.4f %s \n %s %i %s %5.4f \n',...
              'create_atoms    ',Type+(Fib.startType-1),... % create atom
              'single ', Fib.PosXAll(i,Type),...
               Fib.PosYAll(i,Type),...
               Fib.PosZAll(i,Type),...
              ' units box ',...
              ' set atom ',...  % set atom ID vx veloX
              ID,...
              ' vx ',...
              Fib.veloX(i,Type));
          ID =ID+1;

          else % Output without set velocities
               fprintf(InitFibPos, '%s %i %s %5.4f %5.4f %5.4f %s \n',...
              'create_atoms    ',Type+(Fib.startType-1),... % create atom
              'single ', Fib.PosXAll(i,Type),...
               Fib.PosYAll(i,Type),...
               Fib.PosZAll(i,Type),...
              ' units box ');
          end
      end
   end
end
disp('.txt-File is done!')

%% ------------------- Plot Fibers at initial position ---------------- %
funcStat=0;
funcStat=func_PlotInitialPos(Fib.nAtomType,Fib,Box);

%% --------------------------- End of Program ------------------------- %
disp('End of program - Have fun...')
```

```matlab
function funcStat=func_PlotInitialPos(nAtomType,Fib,Box);

% Plot initial positions of all fibers in channel with square cross section

global interpreterName resDir figVisible

%% -------------------- Formating ----------------------------------------%
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

%% --------------------  Figure / Plot  ----------------------------------%
% -------------------- Plot yz-Plane ---------------------------------%
figName = 'Initial position of all fibers yz';
scrs = get(0,'screensize');
rect = [10, 10, 600, 500]; %[left, bottom, width, height]

fig1 = figure('Name',figName,'Position',rect,'Visible',figVisible);

% subplot(2,1,1) % yz
for Type=1:1:nAtomType
          plot(Fib.PosYAll(:,Type), Fib.PosZAll(:,Type),...
           symbolArray{Type},...
          'MarkerSize',markersize*Type*0.1);hold on;
     legendText{Type}=...
          [preAfterSymbol,'AR=',num2str(Fib.AR(Type)),preAfterSymbol];
    end

 legend(legendText,'Location','Northwest');

 if(  strcmp(interpreterName, 'latex') )
   legend(legendText,'fontsize',fontSizeLabel*0.85,...
       'interpreter',interpreterName,'Location','Northwest');
 end

    % Lines
    % plot([Box.yMin Box.yMax],[0 0],'k-.','MarkerSize',2);hold on;
    % plot([0 0],[Box.zMin Box.zMax],'k-.','MarkerSize',2);hold off;

    axis([Box.yMin Box.yMax Box.zMin Box.zMax]);
    axis square;
    set(gca,'ZLim',[Box.zMin Box.zMax]);
    set(gca,'YLim',[Box.yMin Box.yMax]);
%     title([preAfterSymbol,'initial fiber distribution',' y^{+}-z^{+}',...
%          preAfterSymbol],...
%         'Interpreter',interpreterName,'fontsize',fontSizeTitle);
    xlabel([preAfterSymbol,'y^{+}',preAfterSymbol],...
        'Interpreter',interpreterName);
    ylabel([preAfterSymbol,'z^{+}',preAfterSymbol],...
        'Interpreter',interpreterName);

    grid off
    box on
    legend boxoff
    set(0,'defaultaxesfontsize',defaultaxesfontsize);
    set(0,'defaulttextfontsize',defaulttextfontsize);
    set(gca,'FontSize',fontSizeAxis);
    set(gca,'FontWeight','normal');
    xlhand = get(gca,'xlabel');
    ylhand = get(gca,'ylabel');
    zlhand = get(gca,'zlabel');
    set(xlhand,'fontsize',fontSizeLabel);
    set(ylhand,'fontsize',fontSizeLabel);
    set(zlhand,'fontsize',fontSizeLabel)
    set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
    set(xlhand,'FontWeight','bold');
    set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
    set(ylhand,'FontWeight','bold');
    set(zlhand,'FontName','Times'); set(zlhand,'FontAngle','italic');
    set(zlhand,'FontWeight','bold');

% -------------------- Save Figure ----------
Name='Particle_InitialPosition_yz';
printAs='-dpdf';
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 23])
print(printAs,'-r450', [resDir,Name])

% -------------------- Plot xy-Plane ---------------------------------%
figName = 'Initial position of all fibers xy';
scrs = get(0,'screensize');
rect = [10, 10, 900, 600]; %[left, bottom, width, height]

fig1 = figure('Name',figName,'Position',rect,'Visible',figVisible);
% subplot(2,1,2) % xy
    for Type=nAtomType:-1:1
         plot(Fib.PosXAll(:,Type), Fib.PosYAll(:,Type),...
           symbolArray{Type},...
          'MarkerSize',markersize*0.25);hold on;
    end
    axis([Box.xMin Box.xMax Box.zMin Box.zMax])
    axis equal
    set(gca,'xLim',[Box.xMin Box.xMax])
    set(gca,'YLim',[Box.zMin Box.zMax])
%     title([preAfterSymbol,'initial fiber distribution','x^{+}-y^{+}',...
%          preAfterSymbol],...
%         'Interpreter',interpreterName,'fontsize',fontSizeTitle);
    xlabel([preAfterSymbol,'x^{+}',preAfterSymbol],...
        'Interpreter',interpreterName);
     ylabel([preAfterSymbol,'y^{+}',preAfterSymbol],...
        'Interpreter',interpreterName);

     grid off
     box on
     legend boxoff
     set(0,'defaultaxesfontsize',defaultaxesfontsize);
     set(0,'defaulttextfontsize',defaulttextfontsize);
     set(gca,'FontSize',fontSizeAxis);
     set(gca,'FontWeight','normal');
     xlhand = get(gca,'xlabel');
     ylhand = get(gca,'ylabel');
     zlhand = get(gca,'zlabel');
     set(xlhand,'fontsize',fontSizeLabel);
     set(ylhand,'fontsize',fontSizeLabel);
     set(zlhand,'fontsize',fontSizeLabel)
     set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
     set(xlhand,'FontWeight','bold');
     set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
     set(ylhand,'FontWeight','bold');
     set(zlhand,'FontName','Times'); set(zlhand,'FontAngle','italic');
     set(zlhand,'FontWeight','bold');

% -------------------- Save Figure ----------
Name='Particle_InitialPosition_xy';
printAs='-dpdf';
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 23])
print(printAs,'-r450', [resDir,Name])

funcStat=1;
end
```

```
Input Data:
=====================================================================
xMin xMax yMin yMax zMin zMax ... simulation domain
xPos ... x-position of all fibres (on one plane)
PartofCS ... part of Cross Section 1 (total), 2(half), 41 (lower quater), 42 (higher quater)
sameDist ... 1 or 2 ... all classes same distance basis shortest or longest fibre, 0 ... all classes same number of particles
StartType ... start Type number in output with ...
setVeloX  . set velocity in x direction for each fibre
rhoFib ... density Fibers
rhoFluid ... density Fluid
nFibY ... number of Fibres in y-direction
nFibZ ... number of Fibres in z-direction
dMajor ... dMajor of Fibre class
dMinor ... dMinor of Fibre class

BOX DIMENSIONS:
===============
xMin    xMax    yMin    yMax    zMin    zMax   xPos  PartofCS sameDist StartType setVeloX
-20.5    6.5     -0.5    0.5    -0.5    0.5    -20.0    2       0         2        1

FIBRE/FLUID DATA & DEFORMATION MODE:
===================================
rhoFib  rhoFluid
1.27    1.0

nFibY   nFibZ
 10       20

dMajor      dMinor
0.040          0.020
0.100          0.020
0.200          0.020
0.300          0.020
0.400          0.020

```

```
%% This preprocessing program provides Fibre Data (position and velocity)
% (e.g for usage in a CFDEM simulation)
% Lisa Koenig, August 2015, TUGraz
%
%  Schematic Diagramm:
% ---------------------
%
%            Z-dir.                       Z-dir.
%              /\                           /\
%              ||                           ||
%              ||                           ||
%          +--------+    +---------------------------------+
%          |        |   |                                  |
%  Y-dir.<- |       |   |                                  | -> X-dir.
%          |        |   |                                  |
%          +--------+    +------------+---------+----------+
%                                     |         |
%                                     |         |
%
% The fibres are positioned on a yz-plane perpendicular to the
% flow direction (x-direction).The fibres can either be spread over:
% (partOfCS=1) the whole corss section,
% (partOfCS=2) half of the cross section,
% (partOfCS=41) in the lower right quater,
% (partOfCS=42) in the upper right quater of the corss section.
%
% Multiple Fibre classes (separated acc. to thei major length, dMajor)
% can be inserted. The positioning of the fibres, which is always limited
% by the required distance to the wall(dMajor/2), since the fibres are
% not allowed to intersect the wall, can be done in 3 different ways
% (0) Distance between fibres in each class is based on each classes
%     dMajor, number of fibres in each class is the same.
% (1) Distance of all Fibres in all classes is given by shortest
%     fibre (class), fibres that don't fit because they are to long
%     and thus to close to the wall are not set. Fibre positions
%     are the same for all fibres. It is just possible that on the
%     wall-near positions there aren't all fibres present.
% (2) Distance of all Fibres in all classes is given by longest fibre
%     (class). Thus all fibre stypes are on all positions.
%     Distance to the wall is given bythe longest fibre.
%     -> Short fibres that would fit close to the wall are still far way.
%
% The Fibre initial velocity can be set with the Velocity profil given by:
% Tamayol, Bahrami - Laminar Flow in Mircochannels With
% Noncircular Cross Section
% setVeloX=0 ... initial fibre velocity is zero
% setVeloX=1 ... initial fibre velocity is calculated

clear all; clc; close all; fclose all
global interpreterName resDir currDir figVisible

%% -------------------- Directories ------------------------------------- %
inDir = '../../input/';
fileName= 'input_Case_1.txt';
resDir='../../preProcessing/';
currDir=pwd;
figVisible='on'

%% -------------------- Read temporary file for setting ----------------- %
% Check if octave of matlab is used
tempDataFile = fopen('tempData.txt');
C = textscan(tempDataFile,'%f %f',1);
fclose(tempDataFile);
InterPreterName=C{1,1};

if InterPreterName == 1 % MATLAB
    interpreterName='latex'
end

if InterPreterName == 2 %OCTAVE
```

```
   interpreterName='tex'
end

%% -------------------- Prepare environment ----------------------------- %
% Check if output directory exists
cd ..
cd ..
if exist('preProcessing') == 0 % if folder does not exist create it!
  mkdir('preProcessing')
end
cd (currDir)

%% -------------------- Read Input Data --------------------------------- %

   % Get Data from input File via dlmread()
   temp.Data1 = dlmread(strcat(inDir,fileName),'',[18 0 18 10]);
   temp.Data2 = dlmread(strcat(inDir,fileName),'',[23 0 23 1]);
   temp.Data3 = dlmread(strcat(inDir,fileName),'',[26 0 26 1]);
   temp.Data4 = dlmread(strcat(inDir,fileName),'',29,0);

   % Box Dimensions and Calculation Option
   Box.xMin = temp.Data1(1);
   Box.xMax = temp.Data1(2);
   Box.yMin = temp.Data1(3);
   Box.yMax = temp.Data1(4);
   Box.zMin = temp.Data1(5);
   Box.zMax = temp.Data1(6);
   Box.xPosAll= temp.Data1(7);
   Box.partOfCS = temp.Data1(8);
   Box.sameDist = temp.Data1(9);
   Fib.startType = temp.Data1(10);
   Fib.setVeloX = temp.Data1(11);

   % Fibre/Fluid Data
   Fib.rho  = temp.Data2(1);
   rhoFluid = temp.Data2(2);

   % Number of Fibres in Z and Y direction
   Fib.nFibY = temp.Data3(1);
   Fib.nFibZ = temp.Data3(2);

   % Fibre Data
   Fib.Data = sortrows(temp.Data4,[1 2]); % Sort acc. to dMajor
   Fib.dMajor = Fib.Data(:,1);
   Fib.dMinor = Fib.Data(:,2);

   clear temp.*; % clear temporary data


%% -------------------- Fibre Values Calculation ------------------------ %
% Diameter of Spherocylinder
Fib.dCylinder =1.24.*Fib.dMinor./sqrt(log(Fib.dMajor./Fib.dMinor));
% Aspect Ratios
Fib.AR        =Fib.dMajor./Fib.dMinor;
% Vol. single fibre in class i; fibre assumed to be ellipsoid
Vol.Fibi      = Fib.dMajor.^3 ./ Fib.AR .^2 .* pi ./ 6;
% mass of one single fibre in class i
mass.Fibi     = Fib.rho * Vol.Fibi;
% vol. equiv. diameter
Fib.dSphere   = (Fib.dMinor.*Fib.dMinor.*Fib.dMajor).^(1/3);

%% -------------------- Fibre Positioning ------------------------------ %
% number of atom types for output file -> CFDEM
nAtomType = length(Fib.AR);

% Number of Fibres depending on the choosen cross section area
   if Box.partOfCS == 1;    % total cross section
      disp('Total cross section');
   elseif Box.partOfCS == 2; % half
      %  Fib.nFibY = Fib.nFibY/2;
      disp('Half cross section');
   elseif or(Box.partOfCS == 41,Box.partOfCS == 42); % quater
      %  Fib.nFibY=Fib.nFibY/2;
      %  Fib.nFibZ=Fib.nFibZ/2;
      disp('Quater cross section');
   else
      error('Box.partOfCS has to be 1, 2, 41, 42');
   end
   Fib.nFibTot=Fib.nFibZ*Fib.nFibY;

% set max./min. y- and z-position for all fibers
% depending on the choosen cross section area
 if Box.partOfCS == 1;    % total cross section
  yMinPos = Box.yMin+Fib.dMajor/2;  yMaxPos = Box.yMax-Fib.dMajor/2;
  zMinPos = Box.zMin+Fib.dMajor/2;  zMaxPos = Box.zMax-Fib.dMajor/2;

 elseif Box.partOfCS == 2; % half the cross section
  yMinPos = ones(length(Fib.dMajor),1)*0; yMaxPos = Box.yMax-Fib.dMajor/2;
  zMinPos = Box.zMin+Fib.dMajor/2;  zMaxPos = Box.zMax-Fib.dMajor/2;

 elseif Box.partOfCS == 41; % high quater of the cross section
  yMinPos = ones(length(Fib.dMajor),1)*0; yMaxPos = Box.yMax-Fib.dMajor/2;
  zMinPos = Box.zMin+Fib.dMajor/2; zMaxPos = ones(length(Fib.dMajor),1)*0;

 elseif Box.partOfCS == 42; % low quater of the cross section
  yMinPos = ones(length(Fib.dMajor),1)*0; yMaxPos = Box.yMax-Fib.dMajor/2;
  zMinPos = ones(length(Fib.dMajor),1)*0; zMaxPos = Box.zMax-Fib.dMajor/2;
  end

 % Fibre Distance in y -direction for all classes
   FibDistY = (abs(yMaxPos-yMinPos))./(Fib.nFibY-1);
 % Fibre Distance in z -direction for all classes
   FibDistZ = (abs(zMaxPos-zMinPos))./(Fib.nFibZ-1);

% Loop over all Fibre Types
for Type=1:1:nAtomType

  % Switch: How are the fibres distributed
  if Box.sameDist == 0     % 0...all classes dist acc. to length
      s = Type;
  elseif Box.sameDist == 1 % 1...all Fib same dist based on shortest fibs
      s = 1;
  elseif Box.sameDist == 2 % 2...all Fib same dist based on longest fib
      s = nAtomType;
  else
      disp('Check sameDist! it can eiter be 0, 1 or 2')
  end

  % Possible Length Intervall in z and y direction
  InterY = [yMinPos(s) yMaxPos(s)];
  InterZ = [zMinPos(s) zMaxPos(s)];

    % Fibres y-directions
    for i=1:1:Fib.nFibY;
     tempPosY= InterY(1)+ FibDistY(s).*(i-1);
        if and(tempPosY >= Box.yMin+Fib.dMajor(Type)/2,...
               tempPosY <= Box.yMax-Fib.dMajor(Type)/2)
          Fib.PosY(i,Type)= tempPosY;
        else
          Fib.PosY(i,Type)=NaN;
        end
    end

    % Fibres z-directions
     for i=1:1:Fib.nFibZ;
     tempPosZ= InterZ(1)+ FibDistZ(s).*(i-1);
        if and(tempPosZ >= Box.zMin+Fib.dMajor(Type)/2,...
               tempPosZ <= Box.zMax-Fib.dMajor(Type)/2);
          Fib.PosZ(i,Type)= tempPosZ;
        else
```

```
            Fib.PosZ(i,Type)=NaN;
          end
        end

        % Combine Positions
        k=1;
        for i=1:1:length(Fib.PosY(:,Type))
          for j=1:1:length(Fib.PosZ(:,Type))
%             Fib.PosYAll(i,j,Type) = Fib.PosY(i,Type);
%             Fib.PosZAll(i,j,Type) = Fib.PosZ(j,Type);
          Fib.PosYAll(k,Type) = Fib.PosY(i,Type);
          Fib.PosZAll(k,Type) = Fib.PosZ(j,Type);
          k=k+1;
          end
        end
        % Fibre x-directions
        Fib.PosXAll=Fib.PosYAll;
        Fib.PosXAll(:,:)=Box.xPosAll;
end

%% -------------------- Set Fibre Velocity ---------------------------- %
% set Inlet Velocity in x direction acc. to Tamayol, Bahrami - Laminar Flow
% in Mircochannels With Noncircular Cross Section
% Set initial velocity
halfLength=0.5;
UmaxSqu=2.08;

if Fib.setVeloX == 1
    for i=1:1:nAtomType
    r = sqrt(Fib.PosYAll(:,i).^2 + Fib.PosZAll(:,i).^2);
    theta = acos(Fib.PosYAll(:,i)./r);
    tempVeloX = velSCS(halfLength, UmaxSqu, r, theta);
    Fib.veloX(:,i)=tempVeloX
      for ii=1:1:size(Fib.veloX(:,i),1)
        % Elimiate NAN velue at center of Channel
        if and(min(Fib.PosYAll(ii,i))==0,min(Fib.PosZAll(ii,i))==0)
           Fib.veloX(ii,i)=UmaxSqu;
        end

        % Elimiate negative Values
         if Fib.veloX(ii,i) < 0
            Fib.veloX(ii,i) =0;
         end
      end
      plot3(Fib.PosYAll(:,1),Fib.PosZAll(:,1),Fib.veloX(:,1),'.');
      hold on;
    end
end

%% -------------------- Write Output File ----------------------------- %
%% Create txt - File
% create_atoms    1 single initialPosX initialPosY1 initialPosZ1 units box
    ID =1;
for Type=1:1:nAtomType
 % Open output file
 InitFibPos=fopen(...
  fullfile(resDir,['in.ParticlePos_AR',num2str(Fib.AR(Type)),'.txt']),'w');
    for i=1:1:size(Fib.PosXAll,1);
       if or(isnan(Fib.PosYAll(i,Type)),isnan(Fib.PosZAll(i,Type)))
       else
         if Fib.setVeloX == 1 % Output with set velocities
         fprintf(InitFibPos,...
              '%s %i %s %5.4f %5.4f %5.4f %s \n %s %i %s %5.4f \n',...
              'create_atoms    ',Type+(Fib.startType-1),... % create atom
              'single ', Fib.PosXAll(i,Type),...
               Fib.PosYAll(i,Type),...
               Fib.PosZAll(i,Type),...
              ' units box ',...
              ' set atom ',...  % set atom ID vx veloX
              ID,...
              ' vx ',...
              Fib.veloX(i,Type));
           ID =ID+1;

           else % Output without set velocities
               fprintf(InitFibPos, '%s %i %s %5.4f %5.4f %5.4f %s \n',...
              'create_atoms    ',Type+(Fib.startType-1),... % create atom
              'single ', Fib.PosXAll(i,Type),...
               Fib.PosYAll(i,Type),...
               Fib.PosZAll(i,Type),...
              ' units box ');
           end
       end
    end
end
disp('.txt-File is done!')

%% -------------------- Plot Fibers at initial position ----------------- %
funcStat=0;
funcStat=func_PlotInitialPos(nAtomType,Fib,Box);

%% ---------------------------- End of Program -------------------------- %
disp('End of program - Have fun...')
```

```
function funcStat=func_PlotInitialPos(nAtomType,Fib,Box);

% Plot initial positions of all fibers in channel with square cross section

global interpreterName resDir figVisible

%% -------------------- Formating ----------------------------------------%
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
   run('formatting_GNU.m');
end

%% --------------------  Figure / Plot  ----------------------------------%
figName = 'Initial yz position of all fibers';
scrs = get(0,'screensize');
rect = [10, 10, 600, 600]; %[left, bottom, width, height]

fig1 = figure('Name',figName,'Position',rect,'Visible',figVisible);

%% -------------------- SubPlot yz-Plane ---------------------------------%
% subplot(2,1,1) % yz
for Type=1:1:nAtomType
          plot(Fib.PosYAll(:,Type), Fib.PosZAll(:,Type),...
           symbolArray{Type},...
          'MarkerSize',markersize*0.5);hold on;
     legendText{Type}=...
          [preAfterSymbol,'AR=',num2str(Fib.AR(Type)),preAfterSymbol];
    end

  legend(legendText,'Location','NorthWest');

  if(  strcmp(interpreterName, 'latex') )
    legend(legendText,'fontsize',fontSizeLabel,...
        'interpreter',interpreterName,'Location','NorthWest');
  end

%     % Lines
%      plot([Box.yMin Box.yMax],[0 0],'k-.','MarkerSize',2);hold on;
%      plot([0 0],[Box.zMin Box.zMax],'k-.','MarkerSize',2);hold off;

    axis([Box.yMin Box.yMax Box.zMin Box.zMax]);
    axis square;
    set(gca,'ZLim',[Box.zMin Box.zMax]);
    set(gca,'YLim',[Box.yMin Box.yMax]);
%     title('initial fiber distribution y-z',...
%         'Interpreter',interpreterName,'fontsize',fontSizeTitle);
    xlabel([preAfterSymbol,'y^{+}',preAfterSymbol],...
        'Interpreter',interpreterName);
    ylabel([preAfterSymbol,'z^{+}',preAfterSymbol],...
        'Interpreter',interpreterName);

    grid off
    box on
    legend boxoff
    axis square
    set(gca,'XTick',[-0.5:0.25:0.5])
    set(gca,'YTick',[-0.5:0.25:0.5])
    set(gca,'XMinorTick','on','YMinorTick','on')
    set(0,'defaultaxesfontsize',defaultaxesfontsize);
    set(0,'defaulttextfontsize',defaulttextfontsize);
    set(gca,'FontSize',fontSizeAxis);
    set(gca,'FontWeight','normal');
    xlhand = get(gca,'xlabel');
    ylhand = get(gca,'ylabel');
    zlhand = get(gca,'zlabel');
    set(xlhand,'fontsize',fontSizeLabel);
    set(ylhand,'fontsize',fontSizeLabel);
    set(zlhand,'fontsize',fontSizeLabel)
    set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
    set(xlhand,'FontWeight','bold');
    set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
    set(ylhand,'FontWeight','bold');
    set(zlhand,'FontName','Times'); set(zlhand,'FontAngle','italic');
    set(zlhand,'FontWeight','bold');


% -------------------- Save Figure ---------------------------------%
Name='Particle_InitialPosition_YZ';
set(gcf, 'paperunits','centimeters',...
     'PaperOrientation','portrait',...
     'paperposition', [0 0 21 18])

printAs='-dpng';
print(printAs,'-r450', [resDir,Name])


%% -------------------- SubPlot xy-Plane ---------------------------------%
figName = 'Initial xz position of all fibers';
scrs = get(0,'screensize');
rect = [10, 10, 400, 900]; %[left, bottom, width, height]

fig2 = figure('Name',figName,'Position',rect,'Visible',figVisible);
% subplot(2,1,2) % xy
   for Type=nAtomType:-1:1
        plot(Fib.PosXAll(:,Type), Fib.PosYAll(:,Type),...
          symbolArray{Type},...
         'MarkerSize',markersize*0.25);hold on;
   end
   axis([Box.xMin Box.xMax Box.zMin Box.zMax])
    axis equal
    set(gca,'xLim',[Box.xMin Box.xMax])
    set(gca,'YLim',[Box.zMin Box.zMax])
    title('initial fiber distribution x-y',...
        'Interpreter',interpreterName,'fontsize',fontSizeTitle);
    xlabel([preAfterSymbol,'x',preAfterSymbol],...
        'Interpreter',interpreterName);
    ylabel([preAfterSymbol,'y',preAfterSymbol],...
        'Interpreter',interpreterName);

    grid off
    box on
    legend boxoff
    set(gca,'XMinorTick','on','YMinorTick','on')
    set(0,'defaultaxesfontsize',defaultaxesfontsize);
    set(0,'defaulttextfontsize',defaulttextfontsize);
    set(gca,'FontSize',fontSizeAxis);
    set(gca,'FontWeight','normal');
    xlhand = get(gca,'xlabel');
    ylhand = get(gca,'ylabel');
    zlhand = get(gca,'zlabel');
    set(xlhand,'fontsize',fontSizeLabel);
    set(ylhand,'fontsize',fontSizeLabel);
    set(zlhand,'fontsize',fontSizeLabel)
    set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
    set(xlhand,'FontWeight','bold');
    set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
    set(ylhand,'FontWeight','bold');
    set(zlhand,'FontName','Times'); set(zlhand,'FontAngle','italic');
    set(zlhand,'FontWeight','bold');

%% -------------------- Save Figure ---------------------------------%
Name='Particle_InitialPosition_XZ';
printAs='-dpng';
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 23])
print(printAs,'-r450', [resDir,Name])

funcStat=1;
end

```

```
% Determine Fibre Orientation of fibres shortly before Side Channel
% especially: get orientation of fibres that get separated
% (c) Lisa Koenig, TU Graz,
%
% Required outMass File:
% ID diameter x y z u v w  (ex ey ez, color)
%  1     2    3 4 5 6 7 8    9 10 11
%
% Required dump:
% 1           ... id
% 2           ... type
% 3 4 5       ... x y z
% 6 7 8       ... vx vy vz
% 9 10 11     ... fx fy fz
% 12 13 14    ...omegax omegay omegaz
% 15          ... radius
% 16 17 18    ... f_ex[1] f_ex[2] f_ex[3]
% 19          ... f_shape[1]
% 20 21 22 23 ...c_cQuatw c_cQuati c_cQuatj c_cQuatk
clear all; close all; clc;

%% Global Parameters
global homeDir resDir inDir FigVisible interpreterName ...
printAs

%% --------------- Global Settings and Environment ---------------------- %
cd('input')
run('Input.m')
cd ..

% % Input
% Re=500;
% dp=0.20;
% SCW=0.5;
% angle=90;
% Linlet=20;
% BuiltIns=0;  % Baffles
%
% % Separation Layer input File
% inFile='CriticalLine_SCW0.5_90_Re_500_dp0.20_xDir.txt';
% inFileDir='input';
%
% % Directories
% homeDir = pwd;
% resDir  = '../../postProcessing/';     % dir for saving calc data,
% inDir       = ['../../outMass/'];      % dir of Input files
% inDirFilePreT  = 'outMassPreT.dat.1';  % preT plane input file
% inDirFilePostT = 'outMassPostT.dat.1'; % post plane input file
% inDirFileSideC = 'outMassSideC.dat.1'; % Side Channel plane input file
%
% FigVisible='off';           % Visible Figures 'on' or 'off'
% printAs='-dpng';            % '-dpdf' (PDF), '-dpng' (PNG)
%
% outSepEff = 'SepEff_Re500_dp0.20_SCW0.5_90.txt'; % output file for Seperation Efficiency
%
% halfLength=0.5;           % Half of the Channel height
% Data.dMinor=0.02;         % dMinor of all Fibres
%
% % Box Dimension Main Channel
% Box.yMin = -0.5; Box.yMax =  0.5;
% Box.zMin = -0.5; Box.zMax =  0.5;

%% ----------------------- Setup Environment -------------------------- %
% Check if output folder ./resDir exists
if exist(resDir) == 0 % if folder does not exist create it!
   mkdir(resDir);
   disp(['Folder: ',fullfile(resDir),' created'])
end

% Check if matlab or octave is used
tempDataFile = fopen('tempData.txt');
C = textscan(tempDataFile,'%f %f',1);
fclose(tempDataFile);
caseIdToRun=C{1,1};
InterPreterName=C{1,2};

if InterPreterName == 1 % MATLAB
   interpreterName='latex';
end

if InterPreterName == 2 %OCTAVE
   interpreterName='tex';
end

%% ----------------------------- Read input --------------------------- %
% PreT (all Fibres in system, doubles possible)
PreTFile = dir(fullfile(inDir,inDirFilePreT));
if PreTFile.bytes <= 45 % only header
 Data.PreT=[] % empty
 disp([inDirFilePreT,' is empty'])
else
 Data.PreT=dlmread(fullfile(inDir,inDirFilePreT),'',1,0); % skip first row
 Data.PreT=sortrows(Data.PreT,1); % sort Data acc. to IDs
end;

% separation layer for ideal orientation hIdeal/(dMajor/2)
Data.SepLayer=dlmread(fullfile(inFileDir,inFile),'',1,0);

%% ------------- Determine general system parameters ------------------ %
Data.ID.All=unique(Data.PreT(:,1));   % Atom IDs
Data.nAtoms=length(Data.ID.All);      % number of Atoms in System
Data.dSphere=unique(Data.PreT(:,2));  % sphere aquiv. diameter
Data.dMajor=round(Data.dSphere.^3./Data.dMinor^2*100)/100; % dMajor axis
Data.AR=Data.dMajor./Data.dMinor;     % aspect ration dMajor/dMinor
Data.nAtomTypes=length(Data.dMajor);  % number of atom types (vary AR)

% Number of fibres in each class/type
for j=1:1:length(Data.dSphere)
   Data.atom_Type(j).nAtoms=0;
   for i=1:1:Data.nAtoms
    Index=find(Data.PreT(:,1)==Data.ID.All(i));
    tempData=Data.PreT(Index(end),2);
        if tempData==Data.dSphere(j)
           Data.atom_Type(j).nAtoms=Data.atom_Type(j).nAtoms+1;
        end
   end
end

% PostT (non-separated fibres, doubles possible)
PostTFile = dir(fullfile(inDir,inDirFilePostT));
if PostTFile.bytes <= 45 % only header
  Data.PostT=[];
  disp([inDirFilePostT,' is empty'])
else
  Data.PostT=dlmread(fullfile(inDir,inDirFilePostT),'',1,0); % skip 1st row
  Data.PostT=sortrows(Data.PostT,1); % sort Data acc. to IDs
end

% SideC (all separated fibres, fibres in side channel)
SideCFile = dir(fullfile(inDir,inDirFileSideC));
if SideCFile.bytes <= 45 % only header
  Data.SideC=[];
  disp([inDirFileSideC,' is empty'])
else
  Data.SideC=dlmread(fullfile(inDir,inDirFileSideC),'',1,0); % skip 1st row
  Data.SideC=sortrows(Data.SideC,1); % sort Data acc. to IDs
end

% Count Passings and get separated IDs
% How many times did each ID pass preT
```

```
Data.ID.PreT(:,1)=Data.ID.All(:,1);
for i=1:1:length(Data.ID.All);
Data.ID.PreT(i,2)=sum(Data.PreT(:,1)==Data.ID.PreT(i,1));
end

% How many times did each ID pass postT
Data.ID.PostT(:,1)=Data.ID.All(:,1);
for i=1:1:length(Data.ID.All);
Data.ID.PostT(i,2)=sum(Data.PostT(:,1)==Data.ID.PostT(i,1));
end

% IDs of separated fibres (fibres in side Channel) after x passings of preT
Data.ID.SideC=Data.SideC(:,1); % IDs of all separated fibres

% Times separated fibres have actually passed preT be4 they got separated
for i=1:1:length(Data.ID.SideC)
   Index=find(Data.ID.PreT(:,1)==Data.ID.SideC(i));
   Data.ID.SideC(i,2)=Data.ID.PreT(Index,2);
end

%% -------------------- Sectioning in Main Channel --------------------- %
[Data]=func_SectioningMainChannel(Box,Data);

%% Devide Section Data in Sections and Atom Types
for j=1:1:Data.nAtomTypes % loop over Types
   for i=1:1:length(Data.Section) % loop over section
      count=1;

      % Fill with NaN
       Data.Section(i).atom_Type(j).atom_data_G(1,:)=...
       ones(1,length(Data.Section(i).atom_data_G(1,:)))*NaN;

      % loop over all particles in section
      for k=1:1:size(Data.Section(i).atom_data_G,1)

       if Data.Section(i).atom_data_G(k,2)==Data.dSphere(j) % diameter
         Data.Section(i).atom_Type(j).atom_data_G(count,:) =...
             Data.Section(i).atom_data_G(k,:);
         count=count+1;
       end

      end
%        plot(Data.Section(i).atom_Type(j).atom_data_G(:,4),...
%            Data.Section(i).atom_Type(j).atom_data_G(:,5),'o');hold on;
%        plot([Box.yMin,Box.yMax],[Box.zMin,Box.zMax],'k-'); hold on;
%        plot([Box.yMin,Box.yMax],[Box.zMax,Box.zMin],'k-');

  % Remove NaN values/rows except if only Nan Values
     if isequal(max(max(Data.Section(i).atom_Type(j).atom_data_G)),NaN)
     Data.Section(i).atom_Type(j).atom_data_G(...
        any(isnan(Data.Section(i).atom_Type(j).atom_data_G),2),:)=[];
     end
   end
end

%% Coordinate Transformation (90 => pi/2) acc. to Section 1-4
% =============================================================
% GLOBAL SYSTEM TO SECTION SYSTEM x -> x''
% Rotation with general rotation matrix D
% Section 1: no transformation needed
% Section 2: transform.  -Pi around x-axis, basis global coord. sys.
%                     orientated like Section 1 coordinate sys.
% Section 3: transform. -2 Pi around x-axis, -||-
% Section 4: transform. -3 Pi around x-axis, -||-

for i=1:1:length(Data.Section) % loop over section
   Data.Section(i).rotAngle=pi/2 .*(i-1);  % rotation angle
   Data.Section(i).rotAxisG2SS=[1 0 0];    % rotation axis (x-axis)

   % General Rotation Matrix
   D=func_GeneralRotationMatrix(Data.Section(i).rotAxisG2SS,...
                              Data.Section(i).rotAngle);

  for j=1:1:Data.nAtomTypes % loop over Types

   Data.Section(i).atom_Type(j).atom_data_SS(1,:)=...
      ones(1,size(Data.Section(i).atom_Type(j).atom_data_G,2))*NaN;

      % loop over all particles in each section and atom type
      for k=1:1:size(Data.Section(i).atom_Type(j).atom_data_G,1)

         % Transfor Data which DOES NOT NEED to change due to rotation
         % ID diameter
         Data.Section(i).atom_Type(j).atom_data_SS(k,1)=...  % ID
             Data.Section(i).atom_Type(j).atom_data_G(k,1);

         Data.Section(i).atom_Type(j).atom_data_SS(k,2)=...  % Diameter
             Data.Section(i).atom_Type(j).atom_data_G(k,2);

         Data.Section(i).atom_Type(j).atom_data_SS(k,3:5)=...
             Data.Section(i).atom_Type(j).atom_data_G(k,3:5); % x y z

         Data.Section(i).atom_Type(j).atom_data_SS(k,6:8)=...
             Data.Section(i).atom_Type(j).atom_data_G(k,6:8); % vx vy vz

%        % No need for transformation!
%        % x y z vx vy vz
%        tempVec2Rot=Data.Section(i).atom_Type(j).atom_data_G(k,3:5).';
%        Data.Section(i).atom_Type(j).atom_data_SS(k,3:5)=...  % x y z
%             (D*tempVec2Rot).';
%
%        tempVec2Rot=Data.Section(i).atom_Type(j).atom_data_G(k,6:8).';
%        Data.Section(i).atom_Type(j).atom_data_SS(k,6:8)=...  % vx vy vz
%             (D*tempVec2Rot).';

         % Transfor Data which DOES change due to rotation
         % ex ey ez
         tempVec2Rot=Data.Section(i).atom_Type(j).atom_data_G(k,9:11).';
         Data.Section(i).atom_Type(j).atom_data_SS(k,9:11)=...% ex ey ez
              (D*tempVec2Rot).';

          % Calc polar angles phi and theta
          %  theta=atan(y''/x'') x'',y'',z'' of orientation vector f_ex
          %  phi=acos(z'')
          xVec=Data.Section(i).atom_Type(j).atom_data_SS(k,3:5);
          if isnan(xVec(1)) % in case there are no values in array
            Data.Section(i).atom_Type(j).atom_data_SS(k,12)=NaN; %theta
            Data.Section(i).atom_Type(j).atom_data_SS(k,13)=NaN; %phi
          else
            [theta,phi]=func_ConvCart2Polar(xVec);
            Data.Section(i).atom_Type(j).atom_data_SS(k,12)=theta;%theta
            Data.Section(i).atom_Type(j).atom_data_SS(k,13)=phi;  %phi
          end
      end
%        plot(Data.Section(i).atom_Type(j).atom_data_SS(:,4),...
%            Data.Section(i).atom_Type(j).atom_data_SS(:,5),'o');hold on;
%        plot([Box.yMin,Box.yMax],[Box.zMin,Box.zMax],'k-'); hold on;
%        plot([Box.yMin,Box.yMax],[Box.zMax,Box.zMin],'k-');
  end
end

%% Data of all Atom Types (from all sections together)
% "leave main channel"
Data.atom_Type(j).atom_data_SS=[];
for i=1:1:length(Data.Section) % loop over section
  for j=1:1:Data.nAtomTypes  % loop over Types
   Data.atom_Type(j).atom_data_SS=...
      vertcat(...
            Data.atom_Type(j).atom_data_SS,...
            Data.Section(i).atom_Type(j).atom_data_SS);
```

```
    end
    Data.atom_Type(j).atom_data_SS=...
        sortrows(Data.atom_Type(j).atom_data_SS,1);
end

%% Data at preT of separated particles passing preT

for j=1:1:Data.nAtomTypes  % loop over Types
    countAll=1; countFst=1;

    Data.atom_Type(j).atom_data_SS_SepFst(1,:)=...
       ones(size(Data.atom_Type(j).atom_data_SS,2),1)*NaN; % at First T-Junc

    Data.atom_Type(j).atom_data_SS_SepAll(1,:)=...
        ones(size(Data.atom_Type(j).atom_data_SS,2),1)*NaN;% at all T-Junc

    for i=1:1:length(Data.ID.SideC) % loop over sep. Fibres
            Index=find(...
                Data.atom_Type(j).atom_data_SS(:,1)==Data.ID.SideC(i,1));

        if not(isempty(Index))
%           % Separated Fibres passing preT only once
%           Data.atom_Type(j).atom_data_SS_SepFst(countFst,:)=...
%               Data.atom_Type(j).atom_data_SS(Index(end),:);
%           countFst=countFst+1;
%
%           % All separated Fibres
%            for ind=1:1:length(Index)
%               Data.atom_Type(j).atom_data_SS_SepAll(countAll,:)=...
%                Data.atom_Type(j).atom_data_SS(Index(ind),:);
%             countAll=countAll+1;
%            end

            % Separated Fibres passing preT only once
            if length(Index)==1
            Data.atom_Type(j).atom_data_SS_SepFst(countFst,:)=...
                Data.atom_Type(j).atom_data_SS(Index,:);
            countFst=countFst+1;
            end

           % All separated Fibres
            Data.atom_Type(j).atom_data_SS_SepAll(countAll,:)=...
                Data.atom_Type(j).atom_data_SS(Index(end),:);
            countAll=countAll+1;
        end
    end
end

%% ------------------------ Angle Distribution ------------------------ %
AngCl=15; % angle classes e.g. 10, 15

% ------------- Separated Fibres passing preT ONLY ONCE!!! -------------
for j=1:1:Data.nAtomTypes % loop over Types
    xAziSepFst(j,:)=-180:AngCl:180;
    xPolSepFst(j,:)=0:AngCl:180;
    yAziSepFst(j,:)=histcounts(...
                Data.atom_Type(j).atom_data_SS_SepFst(:,12).*180./pi,...
                xAziSepFst(j,:)); %theta
    yPolSepFst(j,:)=histcounts(...
        Data.atom_Type(j).atom_data_SS_SepFst(:,13).*180./pi,...
        xPolSepFst(j,:)); %phi

    % Normalize Data (Sum =!= 1) in each fibre class
    yAziNormSepFst(j,:)=yAziSepFst(j,:).*NaN;
    yPolNormSepFst(j,:)=yPolSepFst(j,:).*NaN;
    if not(isequal(sum(yAziSepFst(j,:)),0)); % empty
        yAziNormSepFst(j,:)=yAziSepFst(j,:)./sum(yAziSepFst(j,:));
        yPolNormSepFst(j,:)=yPolSepFst(j,:)./sum(yPolSepFst(j,:));

    end
%    %% Figures
%    NameAzi=['AziAngleDistr_AR',num2str(Data.AR(j))];
%    NamePol=['PolAngleDistr_AR',num2str(Data.AR(j))];
%
%    funcStat=func_plotBarAngleSingle(xAzi(j,:),yAziNorm(j,:),...
%        xPol(j,:),yPolNorm(j,:),Data.AR(j),NameAzi,NamePol)
end

% Figure
%    NameAzi=['AziAngleDistrSepFst_AR',num2str(min(Data.AR)),...
%             '- AR',num2str(max(Data.AR))];
%    NamePol=['PolAngleDistrSepFst_AR',num2str(min(Data.AR)),...
%             '- AR',num2str(max(Data.AR))];
%    funcStat=func_plotBarAngleAll(xAziSepFst,yAziNormSepFst,...
%        xPolSepFst,yPolNormSepFst,Data.AR,NameAzi,NamePol);

    % Figure to compare all Fibre Types orientation STACKED
    % Normalize Data (Sum =!= 1) over all fibre classes
    xAziSepFstStack=xAziSepFst(1,:);
    yAziNormSepFstStack=yAziNormSepFst./nansum(nansum(yAziNormSepFst));
    xPolSepFstStack=xPolSepFst(1,:);
    yPolNormSepFstStack=yPolNormSepFst./nansum(nansum(yPolNormSepFst));

    NameAzi=['AziAngleDistrSepFstStack_AR',num2str(min(Data.AR)),...
             '- AR',num2str(max(Data.AR))];
    NamePol=['PolAngleDistrSepFstStack_AR',num2str(min(Data.AR)),...
             '- AR',num2str(max(Data.AR))];

     funcStat=func_plotBarAngleAllStack(xAziSepFstStack,yAziNormSepFstStack,...
        xPolSepFstStack,yPolNormSepFstStack,Data.AR,NameAzi,NamePol);

% ------------- ALL Separated Fibres passing preT ----------------------
clear xAzi xPol yAzi yPol yAziNorm yPolNorm yAziNormAllStack yPolNormAllStack
for j=1:1:Data.nAtomTypes % loop over Types
    xAziSepAll(j,:)=-180:AngCl:180;
    xPolSepAll(j,:)=0:AngCl:180;
    yAziSepAll(j,:)=histcounts(...
                Data.atom_Type(j).atom_data_SS_SepAll(:,12).*180./pi,...
                xAziSepAll(j,:)); %theta
    yPolSepAll(j,:)=histcounts(...
        Data.atom_Type(j).atom_data_SS_SepAll(:,13).*180./pi,...
        xPolSepAll(j,:)); %phi

    % Normalize Data (Sum =!= 1) in each fibre class
    yAziNormSepAll(j,:)=yAziSepAll(j,:).*NaN;
    yPolNormSepAll(j,:)=yPolSepAll(j,:).*NaN;
    if not(isequal(sum(yAziSepAll(j,:)),0)); % empty
        yAziNormSepAll(j,:)=yAziSepAll(j,:)./sum(yAziSepAll(j,:));
        yPolNormSepAll(j,:)=yPolSepAll(j,:)./sum(yPolSepAll(j,:));

    end
%    %% Figures
%    NameAzi=['AziAngleDistr_AR',num2str(Data.AR(j))];
%    NamePol=['PolAngleDistr_AR',num2str(Data.AR(j))];
%
%    funcStat=func_plotBarAngleSingle(xAzi(j,:),yAziNorm(j,:),...
%        xPol(j,:),yPolNorm(j,:),Data.AR(j),NameAzi,NamePol)
end

% Figure to compare all Fibre Types orientation when they get separated
%    NameAzi=['AziAngleDistrSepAll_AR',num2str(min(Data.AR)),...
%             '- AR',num2str(max(Data.AR))];
%    NamePol=['PolAngleDistrSepAll_AR',num2str(min(Data.AR)),...
%             '- AR',num2str(max(Data.AR))];
%    funcStat=func_plotBarAngleAll(xAziSepAll,yAziNormSepAll,...
%        xPolSepAll,yPolNormSepAll,Data.AR,NameAzi,NamePol);

% Figure to compare all Fibre Types orientation STACKED
    % Normalize Data (Sum =!= 1) over all fibre classes
    xAziSepAllStack=xAziSepAll(1,:);
    yAziNormSepAllStack=yAziNormSepAll./nansum(nansum(yAziNormSepAll));
    xPolSepAllStack=xPolSepAll(1,:);
    yPolNormSepAllStack=yPolNormSepAll./nansum(nansum(yPolNormSepAll));

    NameAzi=['AziAngleDistrSepAllStack_AR',num2str(min(Data.AR)),...
              '- AR',num2str(max(Data.AR))];
    NamePol=['PolAngleDistrSepAllStack_AR',num2str(min(Data.AR)),...
              '- AR',num2str(max(Data.AR))];
      FigVisible='on';
     funcStat=func_plotBarAngleAllStack(xAziSepAllStack,yAziNormSepAllStack,...
          xPolSepAllStack,yPolNormSepAllStack,Data.AR,NameAzi,NamePol);

% -------------- Data of all fibres (investigate orientation of all fibres)
clear xAzi xPol yAzi yPol yAziNorm yPolNorm
for j=1:1:Data.nAtomTypes % loop over Types
    xAziAll(j,:)=-180:AngCl:180;
    xPolAll(j,:)=0:AngCl:180;
    yAziAll(j,:)=histcounts(...
                  Data.atom_Type(j).atom_data_SS(:,12).*180./pi,...
                  xAziAll(j,:)); %theta
    yPolAll(j,:)=histcounts(...
      Data.atom_Type(j).atom_data_SS(:,13).*180./pi,...
      xPolAll(j,:)); %phi

    % Normalize Data (Sum =!= 1) in each fibre class
    yAziNormAll(j,:)=yAziAll(j,:).*NaN;
    yPolNormAll(j,:)=yPolAll(j,:).*NaN;
    if not(isequal(sum(yAziAll(j,:)),0)); % empty
      yAziNormAll(j,:)=yAziAll(j,:)./sum(yAziAll(j,:));
      yPolNormAll(j,:)=yPolAll(j,:)./sum(yPolAll(j,:));

    end
end

% Figure to compare all Fibre Types orientation GROUPED
%     NameAzi=['AziAngleDistrAll_AR',num2str(min(Data.AR)),...
%              '- AR',num2str(max(Data.AR))];
%     NamePol=['PolAngleDistrAll_AR',num2str(min(Data.AR)),...
%              '- AR',num2str(max(Data.AR))];
%     funcStat=func_plotBarAngleAll(xAziAll,yAziNormAll,...
%         xPolAll,yPolNormAll,Data.AR,NameAzi,NamePol);

% Figure to compare all Fibre Types orientation STACKED
    % Normalize Data (Sum =!= 1) over all fibre classes
    xAziAllStack=xAziAll(1,:);
    yAziNormAllStack=yAziNormAll./nansum(nansum(yAziNormAll));
    xPolAllStack=xPolAll(1,:);
    yPolNormAllStack=yPolNormAll./nansum(nansum(yPolNormAll));

%     bar(xAziAllStack, yAziNormAllStack.','stacked')
%     bar(xPolAllStack, yPolNormAllStack.','stacked')

    NameAzi=['AziAngleDistrAllStack_AR',num2str(min(Data.AR)),...
              '- AR',num2str(max(Data.AR))];
    NamePol=['PolAngleDistrAllStack_AR',num2str(min(Data.AR)),...
              '- AR',num2str(max(Data.AR))];
     funcStat=func_plotBarAngleAllStack(xAziAllStack,yAziNormAllStack,...
          xPolAllStack,yPolNormAllStack,Data.AR,NameAzi,NamePol);


%% ---------------- y,z Position in preT before separated --------------- %
% Figure separation at first around
Name='yz-PreT_SepFst';
maxPtsSepFst=0;
for j=1:1:Data.nAtomTypes % loop over Types
    tempmaxPts=length(Data.atom_Type(j).atom_data_SS_SepFst(:,4));
    if tempmaxPts>maxPtsSepFst
      maxPtsSepFst=tempmaxPts;
    end
end
y=ones(maxPtsSepFst,Data.nAtomTypes)*NaN;
z=ones(maxPtsSepFst,Data.nAtomTypes)*NaN;

for j=1:1:Data.nAtomTypes % loop over Types
    for k=1:1:length(Data.atom_Type(j).atom_data_SS_SepFst(:,4))
    y(k,j)=Data.atom_Type(j).atom_data_SS_SepFst(k,4);
    z(k,j)=Data.atom_Type(j).atom_data_SS_SepFst(k,5);
    end
end

funcStat=func_plotYZPositionPreT(y,z,Data.AR,Name);
clear y z

% Figure separation at no matter what around
Name='yz-PreT_SepAll';
maxPtsSepFst=0;
for j=1:1:Data.nAtomTypes % loop over Types
    tempmaxPts=length(Data.atom_Type(j).atom_data_SS_SepAll(:,4));
    if tempmaxPts>maxPtsSepFst
      maxPtsSepFst=tempmaxPts;
    end
end
y=ones(maxPtsSepFst,Data.nAtomTypes)*NaN;
z=ones(maxPtsSepFst,Data.nAtomTypes)*NaN;

for j=1:1:Data.nAtomTypes % loop over Types
    for k=1:1:length(Data.atom_Type(j).atom_data_SS_SepAll(:,4))
    y(k,j)=Data.atom_Type(j).atom_data_SS_SepAll(k,4);
    z(k,j)=Data.atom_Type(j).atom_data_SS_SepAll(k,5);
    end
end

funcStat=func_plotYZPositionPreT(y,z,Data.AR,Name);
clear y z

%% Plot MAX. z+ position of separated fibres
for j=1:1:Data.nAtomTypes % loop over Types
    maxZPos(j)=...
    (halfLength-abs(max(Data.atom_Type(j).atom_data_SS_SepAll(:,5))))...
      /(Data.dMajor(j)/2);
end
Name='maxZPos_preT_SepAll';
funcStat=func_plotZPos(Data.AR,maxZPos,Name,Data.SepLayer);

%% Plot z+ of all separated fibres
maxPts=0;
for j=1:1:Data.nAtomTypes % loop over Types
    tempmaxPts=length(Data.atom_Type(j).atom_data_SS_SepAll(:,5));
    if tempmaxPts>maxPts
      maxPts=tempmaxPts;
    end
end
ZPos=ones(maxPts,Data.nAtomTypes)*NaN;
for j=1:1:Data.nAtomTypes % loop over Types
    for k=1:1:size(Data.atom_Type(j).atom_data_SS_SepAll,1)
    ZPos(k,j)=...
    ((halfLength-abs(Data.atom_Type(j).atom_data_SS_SepAll(k,5)))...
      ./(Data.dMajor(j)/2));
    end
end
Name='ZPos_preT_SepAll';
funcStat=func_plotZPos(Data.AR,ZPos,Name,Data.SepLayer);

%% Separation Efficiency

% --------------- All Fibres
for j=1:1:Data.nAtomTypes % loop over Types
% Number of Fibres in each Class that are separated
Data.atom_Type(j).nAtoms_SepAll=0;

```

```
if not(isnan(max(max(Data.atom_Type(j).atom_data_SS_SepAll))));
  Data.atom_Type(j).nAtoms_SepAll=...
   size(Data.atom_Type(j).atom_data_SS_SepAll,1);
end

% Separation Efficiency of each Class, number based per simulation time
Data.atom_Type(j).SepEffAll=1-...
    Data.atom_Type(j).nAtoms_SepAll/...
    Data.atom_Type(j).nAtoms;
SepEffPlotAll(j)=Data.atom_Type(j).SepEffAll;
end

Name='TeffAll';
funcStat=func_plotSepEff(Data.AR,SepEffPlotAll,Name);

% --------------- All Fibres separated at first time after preT
for j=1:1:Data.nAtomTypes % loop over Types
% Number of Fibres in each Class that are separated
Data.atom_Type(j).nAtoms_SepFst=0;

if not(isnan(max(max(Data.atom_Type(j).atom_data_SS_SepFst))));
  Data.atom_Type(j).nAtoms_SepFst=...
   size(Data.atom_Type(j).atom_data_SS_SepFst,1);
end

% Separation Efficiency of each Class, number based per simulation time
Data.atom_Type(j).SepEffFst=1-...
    Data.atom_Type(j).nAtoms_SepFst/...
    Data.atom_Type(j).nAtoms;
SepEffPlotFst(j)=Data.atom_Type(j).SepEffFst;
end

Name='TeffFst';
funcStat=func_plotSepEff(Data.AR,SepEffPlotFst,Name);

%% Write Output Separation Efficiency
outSepEffFile=fopen(fullfile(resDir,outSepEff),'w');

fprintf(outSepEffFile,'%s \n','Re      dp   Linlet SCW  angle BuiltIns');
fprintf(outSepEffFile,'%5.1f %5.2f %5.2f %5.1f %i \n\n',...
    Re, dp, Linlet, SCW, angle, BuiltIns);
fprintf(outSepEffFile,'%s \n','AR   SepEffAll SepEffFst');
for j=1:1:Data.nAtomTypes % loop over Types
fprintf(outSepEffFile,'%i %f %f \n',...
    Data.AR(j),SepEffPlotAll(j),SepEffPlotFst(j));
end

disp('End of program. Have fun...');

```

```
function funcStat=func_plotBarAngleAllStack(xAzi,yAziNorm,...
        xPol,yPolNorm,AR,NameAzi,NamePol)

global interpreterName printAs resDir FigVisible
% Get Screen Size
%scrsz = get(groot,'ScreenSize');% MATLAB >R2014b
scrsz = get(0,'ScreenSize'); % MATLAB <R2014b

% Formating
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

nTypes=size(AR,1);
barWidthAzi=0.80;
barWidthPol=0.40;

%% Figure Azimuthal Angle
% scrsz = get(0,'screensize');
% FigAzi=figure('Name',NameAzi,'Position',...
%     [1 scrsz(4)/2.2 scrsz(3)/2.5 scrsz(4)/2],'Visible',FigVisible);
rectAzi = [10, 10, 600, 600]; %[left, bottom, width, height]
rectPol = [10, 10, 600, 600]; %[left, bottom, width, height]

FigAzi=figure('Name',NameAzi,'Position',...
    rectAzi,'Visible',FigVisible);

for xi=1:1:length(xAzi)-1
xAziBar(xi)=xAzi(xi)+(xAzi(xi+1)-xAzi(xi))/2;
end

bar(xAziBar,yAziNorm.',...
    'stacked',...
    'barwidth',barWidthAzi);
colormap gray(5); % jet lines

for j=1:1:nTypes
    legendtext{j}=...
        [preAfterSymbol,'AR=',num2str(AR(j)),preAfterSymbol];
end

if(  strcmp(interpreterName, 'tex') )
    legend(legendtext,'Location','Northwest')

    xlabel([preAfterSymbol,'\theta',preAfterSymbol,' - azimuthal angle'],...
          'FontSize',fontSizeLabel);
    ylabel([preAfterSymbol,'\Delta Q_0',preAfterSymbol],...
          'FontSize',fontSizeLabel);
end

if(  strcmp(interpreterName, 'latex') )
   legend(legendtext,'Location','Northwest',...
   'interpreter',interpreterName,...
   'FontSize',fontSizeLabel,'Orientation','vertical');

   xlabel([preAfterSymbol,'\theta',preAfterSymbol,' - azimuthal angle'],...
       'interpreter',interpreterName,'FontSize',fontSizeLabel);
   ylabel([preAfterSymbol,'\Delta Q_0',preAfterSymbol],...
       'interpreter',interpreterName,'FontSize',fontSizeLabel);
end

% Every 3th entry is a labeled xtick
XTickLabelAzi=ones(1,length(xAzi))*NaN;
XTickLabelAzi(1:3:end)=xAzi(1:3:end);
XTickLabelAzi=num2cell(XTickLabelAzi);
% Replace NaN with blank space
XTickLabelAzi(cellfun(@(x) any(isnan(x)),XTickLabelAzi)) = {''};

grid off;
```

```
box on;
legend boxoff;
xlim([-180-10 180+10]);
set(gca,'XTick',[-180:15:180])
set(gca,'XTickLabel',XTickLabelAzi)
% set(gca,'XMinorTick','on','YMinorTick','on')
% set(gca,'XTickLabelRotation',45)
set(0,'defaultaxesfontsize',stdTextFontSize);
set(0,'defaulttextfontsize',stdTextFontSize);
set(gca,'FontSize',stdTextFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',fontSizeLabel);
set(ylhand,'fontsize',fontSizeLabel);
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters');

% Save Figure
set(gcf, 'paperunits','centimeters',...
    'PaperOrientation','portrait',...
    'paperposition', [0 0 21 18])
print(printAs,'-r400', [resDir,NameAzi]);


%% Figure Polar Angle
% scrsz = get(0,'screensize');
% FigPol=figure('Name',NamePol,'Position',...
%     [1 scrsz(4)/2.2 scrsz(3)/2.5 scrsz(4)/2],'Visible',FigVisible);

FigPol=figure('Name',NamePol,'Position',...
    rectPol,'Visible',FigVisible);

for xi=1:1:length(xPol)-1
xPolBar(xi)=xPol(xi)+(xPol(xi+1)-xPol(xi))/2;
end

bar(xPolBar,yPolNorm.',...
    'stacked',...
    'barwidth',barWidthPol,...
    'EdgeColor','k');
colormap gray(5); % jet lines

for j=1:1:nTypes
legendtext{j}=[preAfterSymbol,'AR=',num2str(AR(j)),preAfterSymbol];
end

if(  strcmp(interpreterName, 'tex') )
legend(legendtext,'Location','Northwest');

xlabel([preAfterSymbol,'\phi',preAfterSymbol,' - polar angle'],...
    'FontSize',fontSizeLabel)
ylabel([preAfterSymbol,'\Delta Q_0',preAfterSymbol],...
    'FontSize',fontSizeLabel)
end

if(  strcmp(interpreterName, 'latex') )
  legend(legendtext,'Location','Northwest',...
  'interpreter',interpreterName,...
  'FontSize',fontSizeLabel,'Orientation','vertical');

  xlabel([preAfterSymbol,'\phi',preAfterSymbol,' - polar angle'],...
   'interpreter',interpreterName,'FontSize',fontSizeLabel)
  ylabel([preAfterSymbol,'\Delta Q_0',preAfterSymbol],...
   'interpreter',interpreterName,'FontSize',fontSizeLabel)
end

% Every 3th entry is a labeled xtick
XTickLabelPol=ones(1,length(xPol))*NaN;
XTickLabelPol(1:3:end)=xPol(1:3:end);
XTickLabelPol=num2cell(XTickLabelPol);
% Replace NaN with blank space
XTickLabelPol(cellfun(@(x) any(isnan(x)),XTickLabelPol)) = {''};

grid off;
box on;
legend boxoff;
xlim([-10 180+10]);

set(gca,'XTick',[0:15:180])
set(gca,'XTickLabel',XTickLabelPol)
% set(gca,'XMinorTick','on','YMinorTick','on')
% set(gca,'XTickLabelRotation',45)
set(0,'defaultaxesfontsize',stdTextFontSize);
set(0,'defaulttextfontsize',stdTextFontSize);
set(gca,'FontSize',stdTextFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',fontSizeLabel); set(ylhand,'fontsize',fontSizeLabel)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

% Save Figure
set(gcf, 'paperunits','centimeters',...
    'PaperOrientation','portrait',...
    'paperposition', [0 0 21 18])
print(printAs,'-r400', [resDir,NamePol])

funcStat=1;
end
```

```matlab
function funcStat=func_plotBarAngle(xAzi,yAziNorm,...
        xPol,yPolNorm,AR,NameAzi,NamePol)
global interpreterName printAs resDir FigVisible
% Get Screen Size
%scrsz = get(groot,'ScreenSize');% MATLAB >R2014b
scrsz = get(0,'ScreenSize'); % MATLAB <R2014b

% Formating
run('formatting_MATLAB');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

%% Figure Azimuthal Angle
FigAzi=figure('Name',NameAzi,'Position',...
    [1 scrsz(4)/2.2 scrsz(3)/2.5 scrsz(4)/2],'Visible',FigVisible);
barWidth=0.8

bar(xAzi,yAziNorm.',barWidth,'stacked')
legendtext=['AR=',num2str(AR)]

legend(legendtext,'Location','northeast','Orientation','vertical')

 if(  strcmp(interpreterName, 'latex') )
   legend(legendtext,'Location','northeast',...
   'interpreter',interpreterName,...
   'FontSize',fontSizeLabel,'Orientation','vertical')
 end

xlabel([preAfterSymbol,'\theta',preAfterSymbol,' - azimuthal angle'],...
        'interpreter',interpreterName,'FontSize',fontSizeLabel);
ylabel([preAfterSymbol,'\Delta Q_0',preAfterSymbol],...
        'interpreter',interpreterName,'FontSize',fontSizeLabel);
colormap(gray);

grid off;
box on;
xlim([-180-10 180+10])
set(gca,'XTick',[-180:45:180])
set(0,'defaultaxesfontsize',stdTextFontSize);
set(0,'defaulttextfontsize',stdTextFontSize);
set(gca,'FontSize',stdTextFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',fontSizeLabel); set(ylhand,'fontsize',fontSizeLabel)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

% Save Figure
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 18])
print(printAs,'-r450', [resDir,NameAzi])


%% Figure Polar Angle
FigPol=figure('Name',NamePol,'Position',...
    [1 scrsz(4)/2.2 scrsz(3)/2.5 scrsz(4)/2],'Visible',FigVisible);
barWidth=0.8

bar(xPol,yPolNorm.',barWidth,'stacked')
legendtext=['AR=',num2str(AR)]

legend(legendtext,'Location','northeast','Orientation','vertical')

 if(  strcmp(interpreterName, 'latex') )
   legend(legendtext,'Location','northeast',...
   'interpreter',interpreterName,...
   'FontSize',fontSizeLabel,'Orientation','vertical')
 end

xlabel([preAfterSymbol,'\phi',preAfterSymbol,' - polar angle'],...
    'interpreter',interpreterName,'FontSize',fontSizeLabel)
ylabel([preAfterSymbol,'\Delta Q_0',preAfterSymbol],...
    'interpreter',interpreterName,'FontSize',fontSizeLabel)
colormap(gray)

grid off;
box on;
xlim([-180-10 180+10])
set(gca,'XTick',[-180:45:180])
set(0,'defaultaxesfontsize',stdTextFontSize);
set(0,'defaulttextfontsize',stdTextFontSize);
set(gca,'FontSize',stdTextFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',fontSizeLabel); set(ylhand,'fontsize',fontSizeLabel)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

% Save Figure
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 18])
print(printAs,'-r450', [resDir,NamePol])

funcStat=1;
end
```

```
% Plot separation efficiency per simulation time
function funcStat=func_plotSepEff(AR,SepEff,Name)

global interpreterName printAs resDir FigVisible
% Get Screen Size
%scrsz = get(groot,'ScreenSize');% MATLAB >R2014b
scrsz = get(0,'ScreenSize'); % MATLAB <R2014b
rect = [10, 10, 700, 500]; %[left, bottom, width, height]

% Formating
run('formatting_MATLAB');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

FigSepYX=figure('Name',Name,'Position',...
    rect,'Visible',FigVisible);

% for j=1:1:length(AR)

    plot(AR(:),...
         SepEff(:),...
         symbolArrayCS{1},...
         'LineWidth',2); hold on;

%    legendtextmaxZPreT{j}=...
%     [preAfterSymbol,'AR=',num2str(AR(j)),preAfterSymbol];
% end

xlabel([preAfterSymbol,'AR',preAfterSymbol],...
       'interpreter',interpreterName,'FontSize',fontSizeLabel);
ylabel([preAfterSymbol,'T_{sep}(AR)',preAfterSymbol,' / ',...
        preAfterSymbol,'t^{+}_{DEM}',preAfterSymbol],...
       'interpreter',interpreterName,'FontSize',fontSizeLabel);

% legend(legendtextmaxZPreT,'Location','northeast','Orientation','vertical');

%  if(  strcmp(interpreterName, 'latex') )
%    legend(legendtextmaxZPreT,'Location','northeast',...
%    'interpreter',interpreterName,...
%    'FontSize',fontSizeLabel,'Orientation','vertical');
%  end

axis([0 22 .70 1])
%axis square

grid off;
box on;
% axis square;
set(gca,'XTick',[2 5 10 15 20]);
set(gca,'YTick',[0.7:0.05:1.0]);
set(gca,'XMinorTick','on','YMinorTick','on')
set(0,'defaultaxesfontsize',stdTextFontSize);
set(0,'defaulttextfontsize',stdTextFontSize);
set(gca,'FontSize',stdTextFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',fontSizeLabel); set(ylhand,'fontsize',fontSizeLabel);
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters');

% Save Figure
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 18]);
print(printAs,'-r450', [resDir,Name]);

funcStat=1;
end
```

```
% Plot yz-position at preT
function funcStat=func_plotYZPositionPreT(y,z,AR,Name)

global interpreterName printAs resDir FigVisible
% Get Screen Size
%scrsz = get(groot,'ScreenSize');% MATLAB >R2014b
scrsz = get(0,'ScreenSize'); % MATLAB <R2014b
rect = [10, 10, 600, 600]; %[left, bottom, width, height]

% Formating
run('formatting_MATLAB');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

FigSep=figure('Name',Name,'Position',...
    rect,'Visible',FigVisible);

for j=1:1:size(y,2) % loop over Types
    plot(...
    y(:,j),...
    z(:,j),...
    symbolArray{j}); hold on;

    % Number of points excluding NaN
    nPnts=0;
    for m=1:1:size(y(:,j),1)
        if not(isnan(y(m,j)))
           nPnts=nPnts+1;
        end
    end

    legendtextyzPreT{j}=...
    [preAfterSymbol,'AR=',num2str(AR(j)),preAfterSymbol,' '...
     preAfterSymbol,'(',num2str(nPnts,'%02d'),')',preAfterSymbol];

end


if(  strcmp(interpreterName, 'tex') )
legend(legendtextyzPreT,'Location','northeast','Orientation','vertical');
   xlabel([preAfterSymbol,'y^{+}',preAfterSymbol],...
       'FontSize',fontSizeLabel);
   ylabel([preAfterSymbol,'z^{+}',preAfterSymbol],...
       'FontSize',fontSizeLabel);
end

if(  strcmp(interpreterName, 'latex') )
   legend(legendtextyzPreT,'Location','NorthEast',...
   'interpreter',interpreterName,...
   'FontSize',fontSizeLabel,'Orientation','vertical');

   xlabel([preAfterSymbol,'y^{+}',preAfterSymbol],...
       'interpreter',interpreterName,'FontSize',fontSizeLabel);
   ylabel([preAfterSymbol,'z^{+}',preAfterSymbol],...
       'interpreter',interpreterName,'FontSize',fontSizeLabel);
end


axis([0 0.5 -0.5 0.0])
% axis square;
grid off;
box on;
legend boxoff;
set(gca,'XTick',[0:0.1:0.5]);
set(gca,'YTick',[-0.5:0.1:0.0]);
set(gca,'XDir','reverse');
set(gca,'XMinorTick','on','YMinorTick','on');
set(0,'defaultaxesfontsize',stdTextFontSize);
set(0,'defaulttextfontsize',stdTextFontSize);
```

```
% Plot maximum z+ position of each fibre type at preT

function funcStat=func_plotZPosPos(AR,ZPos,Name,SepLayer)

global interpreterName printAs resDir FigVisible
% Get Screen Size
%scrsz = get(groot,'ScreenSize');% MATLAB >R2014b
scrsz = get(0,'ScreenSize'); % MATLAB <R2014b
rect = [10, 10, 600, 600]; %[left, bottom, width, height]

% Formating
run('formatting_MATLAB');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

FigSepYX=figure('Name',Name,'Position',...
    rect,'Visible',FigVisible);

for j=1:1:length(AR)
    ARPlot(:,j)=ones(size(ZPos(:,j),1),1)*AR(j);

    plot(ARPlot(:,j),...
         ZPos(:,j),...
         symbolArray{j}); hold on;

    % Number of points excluding NaN
    nPnts=0;
    for m=1:1:size(ZPos(:,j),1)
        if not(isnan(ZPos(m,j)))
           nPnts=nPnts+1;
        end
    end

    legendtextZPosPreT{j}=...
    [preAfterSymbol,'AR=',num2str(AR(j)),preAfterSymbol,' '...
     preAfterSymbol,'(',num2str(nPnts,'%02d'),')',preAfterSymbol];

end

% Plot ideal separation layer hideal+/(dMajor/2)
plot(AR(:),SepLayer(:,7)./SepLayer(:,9),'r.-',...
    'markersize',markersize*2); hold on;
legendtextZPosPreT{end+1}=...
    ['ideal sep. layer'];
%     [preAfterSymbol,'h^{+}_{ideal}/(d_{Major,AR}/2)',preAfterSymbol];

% Plot Straight Line at 1
plot([0 22],[1 1],'k-'); hold on;

 if(  strcmp(interpreterName, 'tex') )
    legend(legendtextZPosPreT,'Location','northeast',...
        'Orientation','vertical');
    xlabel([preAfterSymbol,'AR',preAfterSymbol],...
           'FontSize',fontSizeLabel);
    ylabel([preAfterSymbol,...
        '(s-|z_{sep,preT}^{+}|)/(d_{Major,AR}/2)',...
            preAfterSymbol],...
            'FontSize',fontSizeLabel);
%     ylabel([preAfterSymbol,...
%         '\frac{s-|z_{sep,preT}^{+}|}{d_{Major,AR}/2}',...
%             preAfterSymbol],...
%             'FontSize',fontSizeLabel);
 end

 if(  strcmp(interpreterName, 'latex') )
   legend(legendtextZPosPreT,'Location','NorthEast',...
   'interpreter',interpreterName,...
   'FontSize',fontSizeLabel,'Orientation','vertical');

set(gca,'FontSize',stdTextFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',fontSizeLabel); set(ylhand,'fontsize',fontSizeLabel);
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters');

% Save Figure
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 18]);
print(printAs,'-r300', [resDir,Name]);

funcStat=1;
end
```

```
   xlabel([preAfterSymbol,'AR',preAfterSymbol],...
         'interpreter',interpreterName,'FontSize',fontSizeLabel);

   ylabel([preAfterSymbol,'(s-|z_{sep,preT}^{+}|)/(d_{Major,AR}/2)',...
          preAfterSymbol],...
         'interpreter',interpreterName,'FontSize',fontSizeLabel);
%  ylabel([preAfterSymbol,'\frac{s-|z_{sep,preT}^{+}|}{d_{Major,AR}/2}',...
%          preAfterSymbol],...
%         'interpreter',interpreterName,'FontSize',fontSizeLabel);
 end

axis([0 22 0 15])

grid off;
box on;
legend boxoff
% axis square;
set(gca,'XTick',[2 5 10 15 20]);
% set(gca,'YTick',[-0.5:0.25:0.5]);
set(gca,'XMinorTick','on','YMinorTick','on')
set(0,'defaultaxesfontsize',stdTextFontSize);
set(0,'defaulttextfontsize',stdTextFontSize);
set(gca,'FontSize',stdTextFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',fontSizeLabel); set(ylhand,'fontsize',fontSizeLabel);
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters');

% Save Figure
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 21 18]);
print(printAs,'-r300', [resDir,Name]);

funcStat=1;
end

```

```
% Divide fibres into Sections
% (c) Lisa Koenig, Sept 2015
%         +-------------+
%         |\    3    /|
%         | \      / |
%         |   \  /   |
%         | 2   o   4 |
%         |   /  \   |
%         | /      \ |
%          |/    1    /|
%          +-------------+
% input: Data
% PreT-data is Sectioned

function Data=func_SectioningMainChannel(Box,Data);

Data.Section(1).Count=0; Data.Section(2).Count=0;
Data.Section(3).Count=0; Data.Section(4).Count=0;

Data.Section(1).atom_data_G(1,:)=ones(1,length(Data.PreT(1,:)))*NaN;
Data.Section(2).atom_data_G(1,:)=ones(1,length(Data.PreT(1,:)))*NaN;
Data.Section(3).atom_data_G(1,:)=ones(1,length(Data.PreT(1,:)))*NaN;
Data.Section(4).atom_data_G(1,:)=ones(1,length(Data.PreT(1,:)))*NaN;

for i=1:1:size(Data.PreT,1)

   yP(i)=Data.PreT(i,4);  % y-position of particle
   zP(i)=Data.PreT(i,5);  % z-position of particle

   % Sector 1 with (0,0) is in Section 1 due to order
     if or(Box.yMin <   yP(i) & yP(i) <= 0 & ...
           Box.zMin <=  zP(i) & zP(i) < -abs(yP(i)),...          % or
           0        <=  yP(i) & yP(i) <= Box.yMax & ...
           Box.zMin <=  zP(i) & zP(i) <= -abs(yP(i)))
        %disp('sec 1');
        Data.Section(1).Count=Data.Section(1).Count+1;
        Data.Section(1).atom_data_G(Data.Section(1).Count,:)=...
            Data.PreT(i,:);

   % Sector 2
     elseif or( 0          <=  yP(i) & yP(i) <= Box.yMax & ...
               -abs(yP(i)) <   zP(i) & zP(i) <= 0 ,...      % or
                0          <=  yP(i) & yP(i) <= Box.yMax & ...
                0          <=  zP(i) & zP(i) <= abs(yP(i)))
            %disp('sec 2');
        Data.Section(2).Count=Data.Section(2).Count+1;
        Data.Section(2).atom_data_G(Data.Section(2).Count,:)=...
            Data.PreT(i,:);

    % Sector 3
     elseif or(0          <=  yP(i) & yP(i) <= Box.yMax & ...
               abs(yP(i)) <   zP(i) & zP(i) <= Box.zMax,...      % or
               Box.yMin   <=  yP(i) & yP(i) <= 0 & ...
               abs(yP(i)) <=  zP(i) & zP(i) <= Box.zMax)
            %disp('sec 3');
        Data.Section(3).Count=Data.Section(3).Count+1;
        Data.Section(3).atom_data_G(Data.Section(3).Count,:)=...
            Data.PreT(i,:);

     % Sector 4
     elseif or(Box.yMin  <=  yP(i) & yP(i) <= 0 & ...
               0         <=  zP(i) & zP(i) < abs(yP(i)),...        % or
               Box.yMin  <=  yP(i) & yP(i) <= 0 & ...
              -abs(yP(i)) <=  zP(i) & zP(i) <= 0)
            %disp('sec 4');
        Data.Section(4).Count=Data.Section(4).Count+1;
        Data.Section(4).atom_data_G(Data.Section(4).Count,:)=...
            Data.PreT(i,:);
     end
end
```

```

end
```

```
function [theta,phi]=func_ConvCart2Polar(xVec)
% [theta,phi]=func_ConvCart2Polar(xVec)
% (c) Lisa KÃ¶nig
% Function to convert cartasian coordinates to polar coordinates

x = xVec(1);
y = xVec(2);
z = xVec(3);

% Calc theta ... azimuthal angle
% differ cases
if      x > 0
        theta=atan(y/x);
elseif  and (x < 0 , y >= 0)
        theta=atan(y/x)+pi;
elseif  and (x < 0 , y < 0)
        theta=atan(y/x)-pi;
elseif  and (x == 0 , y > 0)
        theta=pi/2;
elseif  and (x == 0 , y < 0)
        theta=-pi/2;
elseif  and (x == 0 , y == 0) % in exact z-direction -> mathe. undfefined
        theta=pi/2;
end

% Calc phi ... polar angle
phi=acos(z);

end
```

```
function [D]=func_GeneralRotationMatrix(rotVec,alpha)
%[D]=func_GeneralRotationMatrix(rotVec,alpha)
% (c) Lisa Maria Koenig, April 2015

% Calculate the general rotation Matrix (germ. Allgemeine Drehmatrix, in
% Matrixform)
% D       ... general rotation matrix
% rotVec ... unit rotation vector
% alpha  ... rotation angle in rad

% Note
% To rotate a vector (aVec) in a coordinate system:
% aVec' = D * aVec
%
% To change basis and get vector (aVec) in new basis coordinates
% (rotate coordinate system, vector doesn't change/rotate!)
% aVec_newBasis = transpose(D) * aVec_oldBasis
%
% This function is an implementation of the general rotation matrix from
% the source: Book: Otto - Rechenmethoden fuer Studierende der Pysik im
% ersten Jahr p.79
%

% General rotation matrix in matrix format
D = [ cos(alpha)      + (1-cos(alpha)) * rotVec(1)^2 ...
      (1-cos(alpha)) * rotVec(1) * rotVec(2) + sin(alpha) * rotVec(3)...
      (1-cos(alpha)) * rotVec(1) * rotVec(3) - sin(alpha) * rotVec(2);...

      (1-cos(alpha)) * rotVec(1) * rotVec(2) - sin(alpha) * rotVec(3)...
       cos(alpha)      + (1-cos(alpha)) * rotVec(2)^2 ...
      (1-cos(alpha)) * rotVec(2) * rotVec(3) + sin(alpha) * rotVec(1);...

      (1-cos(alpha)) * rotVec(1) * rotVec(3) + sin(alpha) * rotVec(2)...
      (1-cos(alpha)) * rotVec(2) * rotVec(3) - sin(alpha) * rotVec(1)...
       cos(alpha)      + (1-cos(alpha)) * rotVec(3)^2];

end
```

```
function funcStat=func_plotBarAngleAll(xAzi,yAziNorm,...
        xPol,yPolNorm,AR,NameAzi,NamePol)

global interpreterName printAs resDir FigVisible
% Get Screen Size
%scrsz = get(groot,'ScreenSize');% MATLAB >R2014b
scrsz = get(0,'ScreenSize'); % MATLAB <R2014b

% Formating
run('formatting_MATLAB.m');
if(  strcmp(interpreterName, 'tex') )
  run('formatting_GNU.m');
end

nTypes=size(xAzi,1);
barWidthAzi=0.90;
barWidthPol=0.45;

%% Figure Azimuthal Angle
% scrsz = get(0,'screensize');
% FigAzi=figure('Name',NameAzi,'Position',...
%     [1 scrsz(4)/2.2 scrsz(3)/2.5 scrsz(4)/2],'Visible',FigVisible);
rectAzi = [10, 10, 600, 600]; %[left, bottom, width, height]
rectPol = [10, 10, 600, 600]; %[left, bottom, width, height]

FigAzi=figure('Name',NameAzi,'Position',...
    rectAzi,'Visible',FigVisible);

bar(xAzi(1,:),yAziNorm.',...
    'grouped',...
    'barwidth',barWidthAzi,...
    'EdgeColor','w',...
    'LineWidt',0.1);
colormap lines(5); % jet

for j=1:1:nTypes
    legendtext{j}=...
        [preAfterSymbol,'AR=',num2str(AR(j)),preAfterSymbol];
end

if(  strcmp(interpreterName, 'tex') )
    legend(legendtext,'Location','northwest')

    xlabel([preAfterSymbol,'\theta',preAfterSymbol,' - azimuthal angle'],...
           'FontSize',fontSizeLabel);
    ylabel([preAfterSymbol,'\Delta Q_0',preAfterSymbol],...
           'FontSize',fontSizeLabel);
end

if(  strcmp(interpreterName, 'latex') )
   legend(legendtext,'Location','northwest',...
   'interpreter',interpreterName,...
   'FontSize',fontSizeLabel,'Orientation','vertical');

   xlabel([preAfterSymbol,'\theta',preAfterSymbol,' - azimuthal angle'],...
       'interpreter',interpreterName,'FontSize',fontSizeLabel);
   ylabel([preAfterSymbol,'\Delta Q_0',preAfterSymbol],...
       'interpreter',interpreterName,'FontSize',fontSizeLabel);
 end


grid off;
box on;
legend boxoff;
xlim([-180-2 180+2]);
ylim([0 1]);
set(gca,'XTick',[-180:45:180])
set(gca,'YTick',[0:0.2:1])
set(gca,'XMinorTick','on','YMinorTick','on')
% set(gca,'XTickLabelRotation',45)
```

```
set(0,'defaultaxesfontsize',stdTextFontSize);
set(0,'defaulttextfontsize',stdTextFontSize);
set(gca,'FontSize',stdTextFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',fontSizeLabel);
set(ylhand,'fontsize',fontSizeLabel);
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters');

% Save Figure
set(gcf, 'paperunits','centimeters',...
    'PaperOrientation','portrait',...
    'paperposition', [0 0 21 18])
print(printAs,'-r450', [resDir,NameAzi]);


%% Figure Polar Angle
% scrsz = get(0,'screensize');
% FigPol=figure('Name',NamePol,'Position',...
%     [1 scrsz(4)/2.2 scrsz(3)/2.5 scrsz(4)/2],'Visible',FigVisible);

FigPol=figure('Name',NamePol,'Position',...
   rectPol,'Visible',FigVisible);

bar(xPol(1,:),yPolNorm.',...
    'grouped',...
    'barwidth',barWidthPol,...
    'EdgeColor','w',...
    'LineWidt',0.1);
colormap lines(5); % jet

for j=1:1:nTypes
legendtext{j}=[preAfterSymbol,'AR=',num2str(AR(j)),preAfterSymbol];
end

if(  strcmp(interpreterName, 'tex') )
legend(legendtext,'Location','northwest');

xlabel([preAfterSymbol,'\phi',preAfterSymbol,' - polar angle'],...
    'FontSize',fontSizeLabel)
ylabel([preAfterSymbol,'\Delta Q_0',preAfterSymbol],...
    'FontSize',fontSizeLabel)
end

if(  strcmp(interpreterName, 'latex') )
   legend(legendtext,'Location','northwest',...
   'interpreter',interpreterName,...
   'FontSize',fontSizeLabel,'Orientation','vertical');

   xlabel([preAfterSymbol,'\phi',preAfterSymbol,' - polar angle'],...
    'interpreter',interpreterName,'FontSize',fontSizeLabel)
   ylabel([preAfterSymbol,'\Delta Q_0',preAfterSymbol],...
    'interpreter',interpreterName,'FontSize',fontSizeLabel)
end

grid off;
box on;
legend boxoff;
xlim([-1 180+1]);
ylim([0 1]);
set(gca,'XTick',[0:45:180])
set(gca,'YTick',[0:0.2:1])
set(gca,'XMinorTick','on','YMinorTick','on')

% set(gca,'XTickLabelRotation',45)
set(0,'defaultaxesfontsize',stdTextFontSize);
set(0,'defaulttextfontsize',stdTextFontSize);
set(gca,'FontSize',stdTextFontSize);
set(gca,'FontWeight','normal');
xlhand = get(gca,'xlabel');ylhand = get(gca,'ylabel');
set(xlhand,'fontsize',fontSizeLabel); set(ylhand,'fontsize',fontSizeLabel)
set(xlhand,'FontName','Times'); set(xlhand,'FontAngle','italic');
set(xlhand,'FontWeight','bold');
set(ylhand,'FontName','Times'); set(ylhand,'FontAngle','italic');
set(ylhand,'FontWeight','bold');
set(gcf, 'paperunits', 'centimeters')

% Save Figure
set(gcf, 'paperunits','centimeters',...
    'PaperOrientation','portrait',...
    'paperposition', [0 0 21 18])
print(printAs,'-r450', [resDir,NamePol])

funcStat=1;
end

```

```
% Plot Format used by running programm with Octave/gnuplot
fontSizeTitle=12;
fontSizeAxis=20;
fontSizeLabel=22;

lineWidth=2;
markersize    = 9;
stdTextFontSize  = 16;

defaulttextfontsize = 18;
defaultaxesfontsize = 18;

if(  strcmp(interpreterName, 'tex') )
   graphics_toolkit gnuplot
   set (0, "defaultaxesfontname", "TimesItalic")
   set (0, "defaultaxesfontsize", stdTextFontSize)
   set (0, "defaulttextfontname", "TimesItalic")
   set (0, "defaultTextFontSize", stdTextFontSize)
   preAfterSymbol = '';
end


 % Formating for experimental data (symbols only)
i=1;
symbolArray{i}='kd';i=i+1;
symbolArray{i}='ro';i=i+1;
symbolArray{i}='bo';i=i+1;
symbolArray{i}='k<';i=i+1;
symbolArray{i}='r<';i=i+1;
symbolArray{i}='b<';i=i+1;
symbolArray{i}='kd';i=i+1;
symbolArray{i}='rd';i=i+1;
symbolArray{i}='ko';i=i+1;

i=1;
faceColorArray{i} = 'w';i=i+1;
faceColorArray{i} = 'w';i=i+1;
faceColorArray{i} = 'b';i=i+1;
faceColorArray{i} = 'k';i=i+1;
faceColorArray{i} = 'r';i=i+1;
faceColorArray{i} = 'k';i=i+1;
faceColorArray{i} = 'b';i=i+1;
faceColorArray{i} = 'w';i=i+1;
faceColorArray{i} = 'w';i=i+1;
faceColorArray{i} = 'w';i=i+1;

% Formatierung der berechneten Daten
i=1;
symbolArrayC{i}='k-';i=i+1;
symbolArrayC{i}='r-';i=i+1;
symbolArrayC{i}='b-';i=i+1;
symbolArrayC{i}='k--';i=i+1;
symbolArrayC{i}='k--';i=i+1;
symbolArrayC{i}='b--';i=i+1;
symbolArrayC{i}='r.-';i=i+1;
symbolArrayC{i}='k.-';i=i+1;
symbolArrayC{i}='b.-';i=i+1;

% Formatierung der berechneten Daten
i=1;
symbolArrayCS{i}='rd-';i=i+1;
symbolArrayCS{i}='ko-';i=i+1;
symbolArrayCS{i}='b<-';i=i+1;
symbolArrayCS{i}='r>--';i=i+1;
symbolArrayCS{i}='k^--';i=i+1;
symbolArrayCS{i}='bv--';i=i+1;
symbolArrayCS{i}='rs.-';i=i+1;
symbolArrayCS{i}='kx.-';i=i+1;
symbolArrayCS{i}='bd.-';i=i+1;

% Formatierung der Linienstaerken
i=1;
lineWidthMultiplicator{i}=1.5; i=i+1;
lineWidthMultiplicator{i}=1.0; i=i+1;
lineWidthMultiplicator{i}=0.5; i=i+1;
lineWidthMultiplicator{i}=1.5; i=i+1;
lineWidthMultiplicator{i}=1.0; i=i+1;
lineWidthMultiplicator{i}=0.5; i=i+1;
lineWidthMultiplicator{i}=1.5; i=i+1;




```

```
% Plot Format used by running programm with Matlab
fontSizeTitle=12;
fontSizeAxis=20;
fontSizeLabel=22;

lineWidth=2;
markersize    = 9;
stdTextFontSize  = 16;

defaulttextfontsize = 18;
defaultaxesfontsize = 18;

preAfterSymbol = '';
if(  strcmp(interpreterName, 'latex') )
    preAfterSymbol = '$';
end


% Formating for experimental data (symbols only)
i=1;
symbolArray{i}='kd';i=i+1;
symbolArray{i}='ro';i=i+1;
symbolArray{i}='bo';i=i+1;
symbolArray{i}='k<';i=i+1;
symbolArray{i}='r<';i=i+1;
symbolArray{i}='b<';i=i+1;
symbolArray{i}='kd';i=i+1;
symbolArray{i}='rd';i=i+1;
symbolArray{i}='ko';i=i+1;

i=1;
faceColorArray{i} = 'w';i=i+1;
faceColorArray{i} = 'w';i=i+1;
faceColorArray{i} = 'b';i=i+1;
faceColorArray{i} = 'k';i=i+1;
faceColorArray{i} = 'r';i=i+1;
faceColorArray{i} = 'k';i=i+1;
faceColorArray{i} = 'b';i=i+1;
faceColorArray{i} = 'w';i=i+1;
faceColorArray{i} = 'w';i=i+1;
faceColorArray{i} = 'w';i=i+1;

% Formatierung der berechneten Daten
i=1;
symbolArrayC{i}='k-';i=i+1;
symbolArrayC{i}='r-';i=i+1;
symbolArrayC{i}='b-';i=i+1;
symbolArrayC{i}='k--';i=i+1;
symbolArrayC{i}='k--';i=i+1;
symbolArrayC{i}='b--';i=i+1;
symbolArrayC{i}='r.-';i=i+1;
symbolArrayC{i}='k.-';i=i+1;
symbolArrayC{i}='b.-';i=i+1;

% Formatierung der berechneten Daten
i=1;
symbolArrayCS{i}='rd-';i=i+1;
symbolArrayCS{i}='ko-';i=i+1;
symbolArrayCS{i}='b<-';i=i+1;
symbolArrayCS{i}='r>--';i=i+1;
symbolArrayCS{i}='k^--';i=i+1;
symbolArrayCS{i}='bv--';i=i+1;
symbolArrayCS{i}='rs.-';i=i+1;
symbolArrayCS{i}='kx.-';i=i+1;
symbolArrayCS{i}='bd.-';i=i+1;

% Formatierung der Linienstaerken
i=1;
lineWidthMultiplicator{i}=1.5; i=i+1;
lineWidthMultiplicator{i}=1.0; i=i+1;
lineWidthMultiplicator{i}=0.5; i=i+1;
lineWidthMultiplicator{i}=1.5; i=i+1;
lineWidthMultiplicator{i}=1.0; i=i+1;
lineWidthMultiplicator{i}=0.5; i=i+1;
lineWidthMultiplicator{i}=1.5; i=i+1;
```

```
function [varargout] = readdump_all(varargin)
% Reads all timesteps from a LAMMPS dump file.
% Input is dump file name with path
% Output is in the form of a structure with following variables
% .timestep     --> Vector containing all time steps
% .Natoms       --> Vector containing number of atoms at each time step
% .x_bound      --> [t,2] array with xlo,xhi at each time step
% .y_bound      --> [t,2] array with ylo,yhi at each time step
% .z_bound      --> [t,2] array with zlo,zhi at each time step
% .atom_data    --> 3 dimensional array with data at each time step stored
%                   as atomdata(:,:,t)
% Example
%       data = readdump_all('dump.LAMMPS');
%
% See also readdump_one, scandump
%
%  Author :  Arun K. Subramaniyan
%           sarunkarthi@gmail.com
%           http://web.ics.purdue.edu/~asubrama/pages/Research_Main.htm
%           School of Aeronautics and Astronautics
%           Purdue University, West Lafayette, IN - 47907, USA.

try
    dump = fopen(varargin{1},'r');
catch
    error('Dumpfile not found!');
end

i=1;
while feof(dump) == 0
    id = fgetl(dump);
     if (strcmpi(id,'ITEM: TIMESTEP'))
            timestep(i) = str2num(fgetl(dump));
    else
     if (strcmpi(id,'ITEM: NUMBER OF ATOMS'))
            Natoms(i) = str2num(fgetl(dump));
     else
      if (strncmpi(id,'ITEM: BOX BOUNDS',15))
            x_bound(i,:) = str2num(fgetl(dump));
            y_bound(i,:) = str2num(fgetl(dump));
            z_bound(i,:) = str2num(fgetl(dump));
      else
       if (strncmpi(id,'ITEM: ATOMS',10))
            for j = 1 : 1: Natoms
                atom_data(j,:,i) = str2num(fgetl(dump));
            end
            i=i+1;
       end
      end
     end
    end
end
%----------Outputs-------------
%OUTPUTS IN SAME VARIABLE STRUCTURE
varargout{1}.timestep = timestep;
varargout{1}.Natoms = Natoms;
varargout{1}.x_bound = x_bound;
varargout{1}.y_bound = y_bound;
varargout{1}.z_bound = z_bound;
varargout{1}.atom_data = atom_data;
%------------------------------
fclose(dump);
```

```
function [ShearRateSqu] = ShearRateSCS(s, Umax, r, theta)

% INPUT
%    m    ...number of sides (4...square)
%    s    ...half side length
%    Umax ...maximal velocity
%    r    ...radial distance
%    theta...angular position of the point
% OUTPUT
% ShearRateSqu  ...the shear rate at the point (r, theta)
% ShearRate SQU = ((du/dr)^2+(du/dtheta)^2)^0.5
%
% Source: S.Radl - Direct Numerical Simulation of Reactive
% Deformable Bubbles in non-Newtonian FluidsDiploma-Thesis
m=4;

A1=0.247+0.767/m^2;
C(1)=(6.01-3.12*m-0.965*m^2)^-1;
if(m==4)
    C(2)=-1.096e-3;
end

eta = r .* tan(pi/m) ./ s;
dudr      = zeros(size(eta,1),size(eta,2));
dudtheta = zeros(size(eta,1),size(eta,2));
for i=1:size(eta,1)
    for j=1:size(eta,2)
        currr = r(i,j);
        currTheta=theta(i,j);
        dudr(i,j)      = -2.*currr.*(tan(pi/m) ./ s)^2./4./A1;
        dudtheta(i,j) = 0;
        for it=1:length(C)
            dudr(i,j)     = dudr(i,j)+ C(it)/A1*m*it.*currr^(m*it-1)*...
                            (tan(pi/m)/s)^(m*it)*cos(m*currTheta);
            dudtheta(i,j)= dudtheta(i,j)+(-C(it)/A1.*currr.^(m*it)*...
                            (tan(pi/m)/s)^(m*it)*m*sin(m*currTheta));
        end
    end
end

dudr = dudr.*Umax;
dudtheta = dudtheta.*Umax;
ShearRateSqu = (dudr.^2+dudtheta.^2).^0.5;
end
```

```matlab
function [vel] = velSCS(s, UmaxSqu, r, theta)
% velDistributionPolygonialDuct - calculates the velocity distribution in a
% polygonial duct. Velocity scaled to give a mean velocity Umean
% Approach based on Tamayol and Bahrami, J Fluids Engineering 132, 111201
% [vel] = velDistributionPolygonialDuct(m, s, Umax, r, theta)
% INPUT
%    m    ...number of sides (4...square)
%    s    ...half side length
%    Umax ...maximal velocity
%    r    ...radial distance
%    theta...angular position of the point
% OUTPUT
%    vel  ...the velocity at the point (r, theta)

m=4;

A1=0.247+0.767/m^2;
C(1)=(6.01-3.12*m-0.965*m^2)^-1;
if(m==4)
    C(2)=-1.096e-3; % S. Radl
end

eta = r .* tan(pi/m) ./ s;
uStar = zeros(size(eta,1),size(eta,2));
for i=1:size(eta,1)
    for j=1:size(eta,2)
        currEta=eta(i,j);
        currTheta=theta(i,j);
        uStar(i,j) = 1-currEta^2/4/A1;
        for it=1:length(C)
            uStar(i,j)=uStar(i,j)+C(it)/A1*currEta^(m*it)*cos(m*currTheta);
        end
    end
end

vel = UmaxSqu*uStar;
end
```