MODELING OF MULTIPHASE FLOW, SUPERHEAT DISSIPATION, PARTICLE TRANSPORT, AND CAPTURE IN CONTINUOUS CASTING OF STEEL

by

Mingyi Liang

© **Copyright by Mingyi Liang, 2022**

All Rights Reserved

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Thermal-Fluids).

Golden, Colorado

Date _____

Signed: _____

Mingyi Liang

Signed: _____

Dr. Brian G. Thomas
Thesis Advisor

Golden, Colorado

Date _____

Signed: _____

Dr. Carl Frick
Professor and Department Head
Department of Mechanical Engineering

ABSTRACT

Over 96% of steel is produced by continuous casting process. During this process, fluid flow in the nozzle and strand can capture detrimental particles such as: inclusions from upstream ladle refining and tundish, entrainment of mold slag, and argon bubbles which are often injected into the nozzle in order to lessen nozzle clogging. In this study, with the aid of existing models, computational fluid dynamics knowledge, and commercial software ANSYS FLUENT, a model system is developed to predict the effect of nozzle design, superheat, and mold-electromagnetic stirrer on multiphase flow in the tundish, nozzle and strand. This model system includes a heat transfer / solidification model, a two-stage bubble-formation model, a Eulerian-Eulerian model of fluid flow, heat transfer in the liquid, a discrete phase model of particle transport, a slag-layer energy-balance model, a Primary Dendrite Arm Spacing model, and particle capture models of hook, entrapment, and engulfment mechanisms. The results match plant nail-board dipping tests of top-surface velocity and ultra-sonic maps of particle capture, both qualitatively and quantitatively. Results suggest that higher superheat lessens the penetration depth of large particles, which leads to less and shallower capture. In the tundish, similar models are developed to investigate the effect of ladle exchange with temperature changes and nozzle misalignment on tundish flow and inclusion exit fraction. Results show that a ladle exchange with lower temperature causes detrimental flow pattern, as the colder and heavier steel tends to flow along the bottom of the hotter steel already in the tundish. This creates a short-cut path for inclusions to exit the tundish rather than being removed into the top slag layer. This makes the steel quality worse, until the tundish liquid eventually cools and the flow pattern returns to its previous state. Nozzle misalignment leads to asymmetrical flow in the tundish, with more large inclusions exiting the far outlet, leading to worse steel quality on that side. This study has practical value, as it provides, for the first time, quantitative insight into the benefits of high superheat / temperature on inclusion-related defects in steel production.

TABLE OF CONTENTS

ABSTRACT iii

LIST OF FIGURES ix

LIST OF TABLES xv

LIST OF SYMBOLS xvi

LIST OF ABBREVIATIONS xxiii

ACKNOWLEDGMENTS xxiv

DEDICATION xxv

CHAPTER 1 INTRODUCTION 1

1.1 Continuous Casting of Steel 1

1.2 Research Motivations and Objectives 3

1.3 Dissertation Outline 4

CHAPTER 2 LITERATURE REVIEW 8

2.1 SEN Design 8

2.2 Superheat in Strand Flow 8

2.3 Particle Capture 9

2.4 Mold-Electromagnetic Stirring 10

2.5 Molten-Steel-Argon-Gas Multiphase Turbulent Model 10

2.6 Tundish Flow 11

CHAPTER 3 MODELING OF MULTIPHASE FLOW WITH TWO DIFFERENT SENS 13

3.1 Abstract 13

3.2 Introduction 13

3.3 Computational Models 14

3.3.1 PFSG and Two-stage Mean Bubble Size Models 14

3.3.2 Heat Transfer and Solidification: CON1D 15

3.3.3 Governing Equations for Multiphase Flow 16

3.4 Flow Model Domain, Mesh, and Boundary Conditions . . . 19
3.5 Nozzle Flow Results . . . 21
3.6 Strand Flow Results . . . 24
3.7 Flow Model Validation with Nail-Board Measurements . . . 25
3.8 Conclusions . . . 26
CHAPTER 4 MODELING OF MULTIPHASE FLOW, SUPERHEAT DISSIPATION, PARTICLE TRANSPORT AND CAPTURE IN A VERTICAL AND BENDING CONTINUOUS CASTER . . . 28
4.1 Abstract . . . 28
4.2 Introduction . . . 28
4.3 Computational Models and Solution Procedure . . . 31
4.3.1 Initial Bubble Size Models: PFSG and Two-Stage Model . . . 32
4.3.2 Heat Transfer and Solidification: CON1D . . . 33
4.3.3 Two-Phase Flow: Eulerian-Eulerian Model . . . 34
4.3.4 Heat Transfer: CFD Thermal Model . . . 36
4.3.5 Energy Balance Model for Heat Flux into Top Slag Layer . . . 37
4.3.6 Lagrangian DPM Model . . . 39
4.3.7 Particle Capture Model: FLUENT UDFs . . . 41
4.3.8 Primary Dendrite Arm Spacing . . . 46
4.4 Flow Model Domain and Mesh . . . 47
4.5 Boundary Conditions . . . 49
4.6 Numerical Details . . . 50
4.7 Post-Processing of Bubble Capture . . . 51
4.8 Casting Conditions . . . 52
4.9 Flow-Model Verification and Validation . . . 53
4.10 Nozzle Flow Results . . . 55
4.11 Strand Flow Results . . . 56
4.12 Temperature distribution Results . . . 61

4.13 Superheat Dissipation Results . . . 62
4.14 Particle Capture Model Validation . . . 62
4.15 Particle Capture Results . . . 65
4.16 Conclusion . . . 73
4.17 Supplement Notes . . . 75
CHAPTER 5 LARGE EDDY SIMULATION FOR MULTIPHASE FLOW IN A STRAIGHT AND BENDING STRAND . . . 77
5.1 Introduction . . . 77
5.2 Computational Models and Solution Procedure . . . 77
5.2.1 Turbulence Model - Large Eddy Simulation . . . 77
5.2.2 Lagrangian DPM Model . . . 78
5.3 Flow Model Domain, Mesh, and Boundary Conditions . . . 79
5.4 Fluid Velocity Results . . . 81
5.5 Nail-board Validation . . . 84
5.6 Conclusions . . . 85
CHAPTER 6 MODELING OF MOLD-ELECTROMAGNETIC STIRRING IN A NOZZLE AND MOLD . . . 86
6.1 Introduction . . . 86
6.2 Computational Models and Solution Procedure . . . 86
6.3 Flow Model Domain, Mesh, and Boundary Conditions . . . 87
6.4 Electromagnetic Force Distribution . . . 89
6.5 Casting Conditions . . . 90
6.6 Hot Argon Volume Flow Rate . . . 91
6.7 Flow Results . . . 92
6.8 Argon Volume Fraction Results . . . 95
6.9 Top Surface Level Results . . . 96
6.10 Particle Transport and Capture Results . . . 97
6.11 Conclusion . . . 100

CHAPTER 7 EFFECT OF LADLE STREAM SUPERHEAT ON TUNDISH FLOW PATTERN AND INCLUSION TRANSPORT AND REMOVAL 102

7.1 Abstract 102

7.2 Introduction 102

7.3 Computational Models and Solution Procedure 104

7.3.1 Governing Equations for Single-Phase Flow 104

7.3.2 Lagrangian DPM Model 105

7.4 Flow Model Domain, Mesh, and Boundary Conditions 107

7.5 Steady-State Temperature Distribution Results 109

7.6 Transient Flow Results- Steel Velocity and Temperature 110

7.7 Inclusion Transport and Exit Fraction Results 113

7.8 Conclusions 117

CHAPTER 8 MODELING OF STEEL FLOW, SUPERHEAT, IMPACT BOX MISALIGNMENT, INCLUSION TRANSPORT, AND REMOVAL IN A TWO-STRAND TUNDISH . . . 119

8.1 Introduction 119

8.2 Computational Models and Solution Procedures 119

8.2.1 Governing Equations for Single-Phase Flow 119

8.2.2 Lagrangian DPM Model 121

8.3 Flow Model Domain and Mesh 122

8.4 Boundary Conditions 122

8.5 Casting Conditions 124

8.6 Steady-State Symmetrical Flow Results 125

8.7 Superheat Simulation Results 125

8.8 Nozzle Misalignment Flow Results 129

8.9 Inclusion Transport and Removal Results 130

8.10 Conclusions 132

CHAPTER 9 SUPPLEMENTAL WORK ON THE ENTRAPMENT / ENGULFMENT USER DEFINED FUNCTION 134

9.1 Introduction 134

9.2 Model Equations and Procedures . . . 135
9.3 Computational Details . . . 136
9.4 Results - Particle Capture . . . 136
9.5 Conclusions . . . 140
CHAPTER 10 CONCLUSIONS AND FUTURE WORK . . . 141
10.1 Modeling System . . . 141
10.2 Conclusions from Simulation Results . . . 142
10.3 Future Work . . . 147
REFERENCES . . . 148
APPENDIX A USER DEFINED FUNCTIONS . . . 156
APPENDIX B COPYRIGHT PERMISSIONS . . . 259
B.1 Copyright Rules . . . 259
B.2 AISTech 2019 Conference Paper . . . 260
B.3 MDPI Journal (Processes) Paper . . . 264
B.4 Permission for Figures . . . 265

LIST OF FIGURES

Figure 1.1 Continuous casting process schematic [3] . . . 2

Figure 1.2 Longitudinal slab casting schematic in SEN and mold region [5] . . . 3

Figure 3.1 Steel shell thickness profiles down the steel strand (a) Narrow face, and (b) Wide face . . . 15

Figure 3.2 Flow domain and mesh for (a) Case 1 nozzle, (b) Case 2 nozzle, and (c) liquid pool . . . 20

Figure 3.3 Boundary conditions for (a) step1: nozzle flow and (b) step2: mold flow . . . 22

Figure 3.4 Steel velocity in the nozzle for (a) Case 1 and (b) Case 2 . . . 22

Figure 3.5 Argon volume fraction and steel flow velocity at SEN ports: (a) Case 1 and (b) Case 2 . . . 23

Figure 3.6 Flow patterns in the strand for (a) Case 1 and (b) Case 2 . . . 24

Figure 3.7 Jet flow in the upper strand for (a) Case 1 and (b) Case 2 . . . 25

Figure 3.8 Top surface velocities comparing CFD predictions and nail board measurements for case 1 . . . 26

Figure 3.9 Top surface velocities comparing CFD predictions and nail board measurements for case 2 . . . 26

Figure 4.1 Model flow chart . . . 31

Figure 4.2 Injection rate and volume fraction of bubbles . . . 33

Figure 4.3 Shell thickness profile: (a) On wide face (WF) (b) On narrow face (NF) . . . 33

Figure 4.4 Slag layer energy balance . . . 38

Figure 4.5 Capture Flow Chart . . . 41

Figure 4.6 Hook capture mechanism schematic . . . 42

Figure 4.7 Capture criterion schematic: (a) Entrapment in between primary dendrites (b) Engulfment by growing solidification shell . . . 44

Figure 4.8 PDAS revolution profile vs. casting time on the wide faces . . . 47

Figure 4.9 Flow domain and mesh (a) Nozzle 3D view, (b) front view, (c) nozzle center plane, side view, (d) Liquid pool, and (e) Liquid pool center plane . . . 48

Figure 4.10 Boundary conditions for (a) Step 1: nozzle flow, (b) Step 2: strand flow, (c) Step3: heat transfer, and (d) Step 4: bubble transport and capture . . . 49

Figure 4.11 Molten steel flow pattern from (a) steady-state RANS results and (b) 38s time-averaged LES results . . . 54

Figure 4.12 Nozzle velocity and outlet argon gas fraction . . . 54

Figure 4.13 Nozzle velocity (a) front view; (b) side view; and (c) argon gas fraction in the plane of the outlet port. . . . 56

Figure 4.14 Close-up view of strand center-plane velocity: (a) 10 K superheat and (b) 30 K superheat case . . . 58

Figure 4.15 Strand center-plane velocity: (a) 10 K superheat and (b) 30 K superheat case . . . 58

Figure 4.16 Modified Froude number distribution for 30 K superheat case: (a) On mold center plane and (b) On symmetry plane. . . . 59

Figure 4.17 Flow pattern on width/4 plane: (a) 10 K superheat, (b) Close-up view, (c) 30 K superheat case, and (d) Close-up view . . . 59

Figure 4.18 10 K superheat case: (a) Steel velocity magnitude and vector 0.15 m from NF, (b) Close-up view; 30 K superheat case: (c) Steel velocity magnitude and vector 0.15 m from NF, (d) Close-up view . . . 60

Figure 4.19 Temperature distribution on center plane across slab width for (a) 10 K superheat case and (b) 30 K superheat case; Temperature distribution near the top surface for (c) 10 K superheat case and (d) 30 K superheat case . . . 61

Figure 4.20 UT measurements (a) Schematic and particle capture band location and (b) Capture band location on the shell thickness evolution profile . . . 63

Figure 4.21 End view – (a) 10 K superheat case: Capture locations of 0.6 mm bubbles and (b) 30 K superheat case: comparison between 0.6mm bubble capture location and UT measurement . . . 64

Figure 4.22 Model validation –number of bubbles captured inside capture band from simulation vs. from UT maps . . . 64

Figure 4.23 End view – Capture locations for different bubbles from 10 K superheat case and 30 K superheat case . . . 66

Figure 4.24 Top view – Escape locations into slag layer for different bubbles from 10 K superheat case and 30 K superheat case . . . 66

Figure 4.25 Front view – Capture locations on WFIR for different bubbles (10 K superheat case) . . . 67

Figure 4.26 Front view – Capture locations on WFIR for different bubbles (30 K superheat case) . . . 67

Figure 4.27 Front view – Capture locations on WFOR for different bubbles (10 K superheat case) . . 68

Figure 4.28 Front view – Capture locations on WFOR for different bubbles (30 K superheat case) . . 68

Figure 4.29 Side view – Capture locations on NF for different bubbles (10 K superheat case) 69

Figure 4.30 Side view – Capture locations on NF for different bubbles (30 K superheat case) 69

Figure 4.31 10 K superheat case – capture and escape fraction of different bubbles . . . 70

Figure 4.32 30 K superheat case – capture and escape fraction of different bubbles 70

Figure 4.33 Capture locations for bubbles by hook capture mechanism for 10 K superheat case on (a) WFIR, (b) WFOR, and (c) NF; for 30 K superheat case on (d) WFIR, (e) WFOR, and (f) NF 71

Figure 4.34 Overall capture fraction of different bubbles, and breakdown by hook, entrapment, and engulfment mechanisms. 72

Figure 4.35 Injection and capture rates for all bubble sizes. 73

Figure 5.1 Geometry and mesh for (a) whole domain (b) nozzle center plane and (c) mold/strand center plane 80

Figure 5.2 Boundary conditions for (a) Eulerian steel phase (b) Heat transfer, (c) Lagrangian bubble transport and capture simulations 81

Figure 5.3 Initial steel flow pattern at t=0s with (a) velocity magnitude contour, (b) velocity vector, (c) close-up view of mold flow 82

Figure 5.4 Flow pattern on center plane at (a) t=0s, (b) t=10s, (c) t=20.56s, (d) t=30s 83

Figure 5.5 Flow pattern on center plane at (a) t=40s, (b) t=50s, (c) t=60s, (d) t=70s 83

Figure 5.6 Time-averaged steel-velocity contour and streamline on center plane with (a) RANS and (b) LES model 84

Figure 6.1 Geometry and mesh for (a) whole domain (b) nozzle center plane and (c) strand center plane 87

Figure 6.2 Boundary conditions for (a) step1: EMS off, (b) step2: EMS on, (c) step3: bubble transport and capture simulations 89

Figure 6.3 Electromagnetic force distribution per unit volume on different planes below meniscus for (a)F_x, (b)F_y, and (c)F_z 90

Figure 6.4 Pressure distribution on domain center plane with EMS off 91

Figure 6.5 Distribution of expansion factor with respect to distance below meniscus 92

Figure 6.6 Flow pattern in the whole domain and close-up view of nozzle with (a) EMS off and (b) EMS on 93

Figure 6.7 Flow pattern near the stopper rod with (a) EMS off and (b) EMS on 93

Figure 6.8 Steel x velocity contour and velocity vectors on nozzle port with (a) EMS off and (b) EMS on 94

Figure 6.9 Steel velocity magnitude and velocity vector with EMS off on planes (a) Y=-0.01m, (b) Y=-0.075m, and (c)Y=-0.15m; with EMS on planes (d) Y=-0.01m, (e) Y=-0.075m, and (f)Y=-0.15m 95

Figure 6.10 Argon volume fraction at center plane with (a) EMS off and (b) EMS on 96

Figure 6.11 Top surface level distribution with (a) EMS off and (b) EMS on 97

Figure 6.12 Top surface level distribution along center lines 97

Figure 6.13 Bubble capture locations with EMS on (a) WF(+z) (b) WF(-z) (c) NF; with EMS off (d) WF(+z) (e) WF(-z) (f) NF 98

Figure 6.14 Bubble capture fraction breakdown, every 0.5m below meniscus with (a) EMS off and (b) EMS on 98

Figure 6.15 Bubble capture percentage by entrapment / engulfment criterion on WF(+z), WF(-z) and NF with and without EM force 99

Figure 6.16 Bubble capture percentage within the hook zone by both entrapment / engulfment criterion and hook mechanism on WF(+z), WF(-z) and NF, with and without EM force 100

Figure 7.1 (a) Geometry of the whole domain (b) mesh of the whole domain (c) mesh near impact box and (d) mesh near dam hole 107

Figure 7.2 Steady-state temperature distribution on (a) symmetry planes, top surface, weir wall, and dam wall, and (b) top surface, impact box wall, weir wall, dam wall, longitudinal wall, and transverse wall 109

Figure 7.3 Flow pattern on center plane at (a) t=40s, (b) t=50s, (c) t=60s, (d) t=70s 110

Figure 7.4 Steel velocity magnitude distribution and vector on longitudinal symmetry plane at t = (a) 0s, (b) 100s, (c) 200s, (d) 1,000s, (e) 1,700s, and (f) 2,700s 112

Figure 7.5 Steel temperature distribution and velocity vector on longitudinal symmetry plane at t = (a) 0s, (b) 100s, (c) 200s, (d) 1,000s, (e) 1,700s, and (f) 2,700s 112

Figure 7.6 Z velocity, V_z, evolution for 1,800s flow time at five monitored points, P1t, P1b, P2t, P2b, and P3 113

Figure 7.7 Number of 10μm inclusions exiting outlet vs. casting time 115

Figure 7.8 Number of 100μm inclusions exiting outlet vs. casting time 115

Figure 7.9 Exit fraction, $C_{f,i}$, for inclusions smaller than and equal to 100μm injected within the first 2500s flow time 116

Figure 7.10 Exit fraction, $C_{f,i}$, for inclusions larger than 100μm injected every 100s interval within the first 2,000s flow time 116

Figure 7.11 Exit fraction of different inclusions 117

Figure 8.1 Computational domain for the two-strand symmetrical tundish 123

Figure 8.2 Computational mesh for the two-strand symmetrical tundish (a) 3-D view and (b) Front view 123

Figure 8.3 Boundary Conditions for (a) fluid flow and (b) particle transport 124

Figure 8.4 Fluid flow in the tundish: (a) Longitudinal front view, (b) Transverse side view, (c) Near the wei-dam region, and (d) Near the impact box region 126

Figure 8.5 The pressure distribution and mass flow rate on tundish outlets 126

Figure 8.6 Tundish flow pattern with iso-thermal, 10 K superheat, and 30 K superheat conditions . 127

Figure 8.7 Temperature distribution on the center plane of tundish 127

Figure 8.8 Super distribution on the tundish inlet, outlet, and walls 128

Figure 8.9 Exit fraction of different inclusions in the 10 K superheat case 128

Figure 8.10 Flow pattern of case M and case S 129

Figure 8.11 3-D view of the streamline in case M 130

Figure 8.12 Inclusion exit and removal fraction at different time 131

Figure 8.13 Overall exit fraction of case M 131

Figure 8.14 Comparison of the exit fraction between case S and case M for different inclusions . . . 132

Figure 9.1 Bubble capture locations on WFIR 137

Figure 9.2 Bubble capture locations on WFOR 137

Figure 9.3 y^+ value at 0.6 mm bubble center 138

Figure 9.4 Bubble capture locations and bubble-center velocity on WFIR 138

Figure 9.5 Capture locations on WFIR for 0.4 mm, 0.5 mm, and 0.6 mm bubbles 139

Figure 9.6 Capture locations on WFOR for 0.4 mm, 0.5 mm, and 0.6 mm bubbles 139

Figure 9.7 Capture locations on WFOR for 0.4 mm, 0.5 mm, and 0.6 mm bubbles 140

Figure B.1 Copy right permission stated on the AIST website 259

Figure B.2 Copy right permission stated on the MDPI website 260

Figure B.3 Copy right permission from the AIST publications manager, Amanda 261

Figure B.4 Copy right permission from the co-author, Xiaoming Ruan 262

Figure B.5 Copy right permission from the co-author, Seong-Mook Cho 262

Figure B.6 Copy right permission from the co-author, Hamed Olia 263

Figure B.7 Copy right permission from the co-author, Lipsa Das 264

Figure B.8 Copy right permission from the co-author, Seong-Mook Cho 265

Figure B.9 Copy right permission from the co-author, Xiaoming Ruan 265

Figure B.10 Figure reprint permission from professor Brian G. Thomas 266

LIST OF TABLES

Table 3.1 Caster dimensions and process conditions 14

Table 3.2 Jet characteristics 23

Table 4.1 Caster dimensions and process conditions 53

Table 6.1 Electromagnetic Force Distribution 89

Table 6.2 Caster Dimensions and Process Conditions for Half of a Caster 90

Table 6.3 Jet characteristics at nozzle ports 94

Table 7.1 Caster Dimensions and Process Conditions 104

Table 7.2 Thermal Boundary Conditions 108

Table 8.1 Thermal Boundary Conditions 124

Table 8.2 Caster Dimensions and Process Conditions 125

LIST OF SYMBOLS

A subscript that denotes s (steel) or g (argon gas) . . . . . q

A uniformly-distributed random number from 0 to 1 . . . . . γ

Additional forces on the argon gas phase . . . . . $\sum F_{ig}$

Additional forces on the steel phase . . . . . $\sum F_{is}$

Ambient temperature . . . . . T_∞

Angle between casting direction and wall-adjacent cell face . . . . . θ_c

Argon gas dynamic viscosity . . . . . μ_g

Argon gas momentum sink term . . . . . $S_{ig-mom-sink}$

Argon gas turbulent viscosity . . . . . μ_{tg}

Argon velocity tensor . . . . . u_{ig}, u_{jg}

Buoyancy force due to liquid / gas density difference . . . . . $F_{Bi,g}$

Cartesian coordinate tensor . . . . . x_i, x_j

Casting speed . . . . . v_c

Casting time . . . . . t_{cas}

Constant for SST k-ω model . . . . . β^*

Constant for SST k-ω model . . . . . σ_{k1}

Constant for SST k-ω model . . . . . σ_ω

Constant for SST k-ω model . . . . . α_ω

Constant for SST k-ω model . . . . . β

Constant for SST k-ω model . . . . . $\sigma_{\omega 2}$

Constant for SST k-ω model . . . . . α_1

Constant for SST k-ω model . . . . . α_2

Constant for SST k-ω model . . . . . β_1

Constant for SST k-ω model . . . . . β_2

Constant for standard k-ϵ*model* . . . . . C_μ

Constant for standard k-ϵ*model* . . . . . $C_{1\epsilon}$

Constant for standard k-ϵ*model* . . . . . $C_{2\epsilon}$

Constants in the drag coefficient (C_{D2}) calculation . . . . . a_1, a_2, a_3

Critical time for a particle to be captured by the hook . . . . . t_{hook}

Cubic root of the cell volume . . . . .

Cumulative time for a particle staying in the hook zone . . . . . t_c

Dendrite tip radius . . . . . r_p

Diffusion coefficient of sulfur in steel . . . . . D_s

Direction tangential to the solidification direction . . . . . η

Distance below meniscus along the strand center . . . . . l_c

Distance between a particle and the closest strand surface . . . . . d_w

Distance between the dendrite tip and the particle . . . . . h_0

Distribution coefficient . . . . . k_1

Downward velocity of slag consumption . . . . . v_y

Drag coefficient for R_{igs} term in the Eulerian-Eulerian model . . . . . C_{D1}

Drag coefficient for drag force calculation in the PDAS model . . . . . C_{D3}

Drag coefficient used in the DPM model . . . . . C_{D2}

Drag force . . . . . F_{iD}

Eddy cross time . . . . . t_{cross}

Eddy length scale . . . . . L_e

Eddy life time . . . . . t_e

Effective thermal conductivity . . . . . k_{eff}

Effective viscosity . . . . . μ_{eff}

Empirical constant for surface tension gradient force . . . . . n_1

Empirical constant for surface tension gradient force . . . . . m

Energy sink term for steel and argon phase . . . . . S_{hs}, S_{hg}

Eotvos number . . . . . E_o

Estimated oscillation frequency for slag consumption calculation . . . . . f

Gas-phase vertical velocity towards the top surface . . . . . v_{gy}

Generation of k . . . . . $G - k$

Gravitational acceleration tensor . . . . . g_i

Gravity / buoyancy force . . . . . F_{iB}

Half of the angle between two dendrite tips and particle center . . . . . θ

Heat capacity of argon gas . . . . . c_{pg}

Heat capacity of liquid slag . . . . . $c_{p,liq}$

Heat capacity of molten steel . . . . . c_{ps}

Heat capacity of sintered slag . . . . . $c_{p,sin}$

Heat capacity of slag power . . . . . $c_{p,pow}$

Heat going into the slag layer from its bottom surface . . . . . $Q_{in,bottom}$

Heat going into the slag layer from its top surface . . . . . $Q_{in,top}$

Heat leaving the slag layer from its top surface . . . . . $Q_{out,top}$

Heat leaving the slag layer from the gap between mold and shell . . . . . $Q_{out,bottom}$

Heat transfer coefficient between slag powder top surface and far-field atmosphere . . . . . h_s

Hook depth . . . . . d_{hook}

Hot molten steel static pressure at argon injection point . . . . . P_h

Hot volume flow rate of argon gas . . . . . Q_h

Interaction time of the eddy and a particle . . . . . t_{int}

Lift force . . . . . F_L

Local temperature for steel and argon phase . . . . . $T_{local,s}$, $T_{local,g}$

Lubrication force . . . . . F_{lub}

Mass flow rate of particle . . . . . $\dot{m}_p$

Mean level fluctuation . . . . . L_F

Minimum distance from the nearest wall . . . . . y

Mixing length for subgrid scales . . . . . L_s

Mixture density . . . . . ρ_m

Mixture turbulent viscosity . . . . . ρ_{tm}

Mixture velocity . . . . . u_m

Mixture viscosity . . . . . μ_m

Modified pressure . . . . . P^*

Momentum source term from discrete particles to continuous phase . . . . . S_{DPM}

Negative strip time . . . . . t_n

Oscillation cycle time . . . . . T

Oscillation frequency used for hook depth calculation . . . . . f_0

Oscillation stroke . . . . . s_0

Particle Reynolds number . . . . . Re_p

Particle density . . . . . ρ_p

Particle diameter . . . . . d_p

Particle radius . . . . . R_p

Particle relaxation time . . . . . τ

Particle velocity tensor . . . . . u_{ip}

Pitch . . . . . h_{hook}

Positive strip time . . . . . t_p

Pressure gradient force . . . . . F_{ip}

Pressure shared by all phase . . . . . P_2

Production limiter . . . . . $\tilde{P}_k$

Projected area of wall-adjacent cell face in casting direction . . . . . A_c

Random velocity fluctuation . . . . . $u_i^{'}$

Reference temperature . . . . . T_{ref}

Sensible energy for steel and argon . . . . . h_s, h_g

Shell thickness . . . . . k

Slab thickness . . . . . t_s

Slab width . . . . . w_s

Slag consumption rate . . . . . C_t

Slag consumption rate per unit length per unit cycle due to lubrication . . . . . c_{lub}

Slag consumption rate per unit length per unit cycle due to oscillation mark . . . . . c_{OM}

Slag density . . . . . ρ_{slag}

Slag liquidus temperature . . . . . $T_{slag,liq}$

Slag solidification temperature for hook depth calculation . . . . . $T_{slag,sol1}$

Slag solidus temperature for slag layer energy balance calculation . . . . . $T_{slag,sol2}$

Slag-powder top-surface temperature . . . . . T_{top}

Smagorinsky constant . . . . . C_s

Solidification constant . . . . . K

Solidification direction . . . . . χ

Solidification velocity . . . . . V_{sol}

Source term for interface forces . . . . . R_{igs}

Sphere volume-equivalent bubble diameter . . . . . d_{eq}

Spread parameter . . . . . n

Standard condition pressure . . . . . P_s

Standard condition temperature . . . . . T_s

Standard condition temperature . . . . . T_s

Standard normally-distributed random number . . . . . ξ

Static pressure . . . . . P

Steel atomic diameter . . . . . a_0

Steel casting temperature . . . . . T_h

Steel density . . . . . ρ_s

Steel dynamic viscosity . . . . . μ_s

Steel dynamic viscosity . . . . . μ_s

Steel kinematic viscosity . . . . . v_s

Steel liquidus temperature . . . . . $T_{steel,liq}$

Steel mass sink term . . . . . $S_{s-mass-sink}$

Steel momentum sink term . . . . . $S_{is-mom-sink}$

Steel thermal diffusion . . . . . α_{th}

Steel turbulent viscosity . . . . . μ_{ts}

Steel velocity tensor . . . . . u_{is}, u_{js}

Strain rate magnitude . . . . . S

Strand cross-sectional area . . . . . A

Subgrid-scale turbulent thermal conductivity . . . . . k_{sgs}

Subgrid-scale turbulent viscosity . . . . . μ_{sgs}

Sulfur content in steel . . . . . C_0

Sulfur content in steel . . . . . C_0

Surface tension between molten steel and argon gas . . . . . σ

Surface tension between steel and slag . . . . . σ_{ss}

Surface tension between steel shell / particle, steel shell / liquid steel, and particle / liquid steel . . . . . σ_{sp}, σ_{sl}, σ_{pl}

Surface tension gradient force . . . . . F_{Grad}

Temperature gradient in front of solidification front . . . . . G_T

The PDAS fitting constant . . . . . λ

The first blending function . . . . . F_1

The second blending function . . . . . F_2

The second blending function . . . . . F_2

Thermal buoyancy force of steel . . . . . $F_{Bi,t}$

Thermal conductivity of argon gas . . . . . k_g

Thermal conductivity of molten steel . . . . . k_s

Time . . . . . t

Time step size for particle trajectory calculation . . . . . t

Top-wall-adjacent cell face area . . . . . A_y

Turbulent Prandtl number . . . . . P_{rL}

Turbulent Prandtl number associated with ϵ . . . . . σ_ϵ

Turbulent Prandtl number associated with k . . . . . σ_k

Turbulent dissipation rate . . . . . ϵ

Turbulent kinetic energy . . . . . k

Turbulent specific dissipation rate . . . . . ω

Virtual mass force . . . . . F_{iVM}

Volume fraction of argon contained in the bubbles which have diameter less than d_{pi} . . . . . $F(d_{pi})$

Volume fraction of argon in a cell . . . . . α_g

Volume fraction of steel in a cell . . . . . α_s

Von karman constant . . . . . κ

Wall normal velocity gradient . . . . . G

Wall-adjacent cell volume . . . . . V_{cell}

Wall-adjacent face area . . . . . A_f

LIST OF ABBREVIATIONS

Alternating Current	AC
Basic Oxygen Furnace	BOFJ
Blast Furnace	BF
Colorado School of Mines	CSM
Computational Fluid Dynamics	CFD
Discrete Phase Model	DPM
Electric Arc Furnace	EAF
Inner Radius	IR
Large Eddy Simulation	LES
Mold-ElectroMagnetic Stirrer	M-EMS
Narrow Face	NF
Outer Radius	OR
Primary Dendrite Arm Spacing	PDAS
Random Walk Model	RWM
Reynolds-Averaged Navior-Stokes	RANS
Shear Stress Transport	SST
Submerged Entry Nozzle	SEN
Ultra-sonic Test	UT
Wide Face	WF

ACKNOWLEDGMENTS

Countless people supported my effort on this dissertation. First and foremost, I would like to express my sincere gratitude to my advisor, Professor Brian G. Thomas for his guidance, support, and encouragement throughout my Ph.D. years. It is my honor to be able to be one of Brian's many students. I learned a great deal from his ever-lasting passion, abundant knowledge, critical way of thinking, and effective communication. Thank you for taking a chance on me, allowing me to making my own mistakes, and always being there to help and keep me on the right track.

I also would like to thank Dr. Seong-Mook Cho, who gave invaluable help, feedback, tips, discussions on my work and helped me find my footing as I started this journey. I feel fortunate to work and learn from him both academically and personally.

I gratefully acknowledge the financial support by the member companies of Continuous Casting Center (CCC) at Colorado School of Mines, which includes: ABB, AK Steel, ANSYS-Fluent, Baosteel, CBMM, Nippon Steel and Sumitomo Metal Corporation, Nucor, RHI Magnesita Refractories, Arcelor-Mittal, SSAB, POSTECH/POSCO, JFE Steel, and Tata Steel. The National Science Foundation (NSF) provides financial support to this work as well.

Over the past five years, I am greatly indebted to many excellent engineers, researchers and scientists who I have been privileged to interact and work with. Dr. Xiaoming Ruan of Baosteel who provided plant blueprints and measurement data; Dr. Sunday Abraham and Dr. Yufeng Wang of SSAB Americas who helped me learn the SSAB water model, sample process and analysis, and Scannig Electron Microscopy (SEM); Rich Gass, Dr. Judy Li of AcelorMittal Indiana-Harbor who had many discussions with Prof. Brian Thomas and me to measure plant device dimension. The support from all of you allows me to complete this project.

I am also greatly appreciative of all the smart scholars and students who I got to learn from. They are: Matthew Zappulla, Hamed Olia, Lipsa Das, Professor Wanming Li, Chang Liu, Yunwei Huang, Ghavam Azizi. Special thanks to Dr. Kai Jin for his previous work on this topic, which lays a solid foundation for me and makes the following work possible.

Thanks to some of my colleagues and friends who helped me during some tough times. Special thanks to Jincheng Lou who helped me in the Thermal-Fluids coursework and Ph.D. qualifying exams.

Lastly, my family deserves endless gratitude. To my family, I give everything, including this.

This thesis is dedicated to my parents, Wenhai Liang and Liyu Wang.

For their endless love, support, and encouragement.

CHAPTER 1
INTRODUCTION

This chapter gives a review and introduction of the continuous casting process of steel. Reasons why this process needs to be studied is provided, and research objectives are listed. A dissertation layout is given at the end as an introduction to the content of each chapter.

1.1 Continuous Casting of Steel

Continuous casting allows molten metal to solidify continuously without interruption by keeping feeding new heats. It is the most cost- and energy-efficient method to mass-produce semifinished metal products with consistent quality in a variety of sizes and shapes [1]. As of 2019 more than 96% of the world's steel production is produced using the continuous casting process [2].

Figure 1.1 (see page 2) shows a schematic of the continuous casting process of steel. Molten steel from the upstream blast furnace (BF) / electric arc furnace (EAF) and Basic Oxygen Furnace (BOF) are transferred into a ladle furnace, where degassing, deoxidation, homogenization, and desulfurization processes are carried out. The molten steel flows from the ladle, through a tundish, into the mold. When running out of molten steel, the old ladle needs to be exchanged with a new molten-steel-filled ladle periodically, and the tundish can hold enough steel to provide continuous flow to the mold. Detrimental inclusions can float up and be removed by the top slag layer in the tundish. Slag layer covers the top of the molten steel in each vessel throughout the continuous casting process to prevent it from exposure to air, which leads to reoxidation and detrimental inclusions. Ceramic nozzles have to be used between vessels to transport the molten steel for the same reason.

Once flowing into the mold through the submerged entry nozzle (SEN), molten steel starts solidifying against the surrounding water-cooled copper-mold walls to form a solid solidifying shell. The top of the shell solidifies against the meniscus, forming subsurface hooks. The shell grows increasingly thick as the molten steel flows downwards. The mold oscillates vertically to prevent the shell from sticking to the mold walls. Supporting drive rolls below the mold anchor the solidifying shell up, which contains the liquid steel inside and allows them to continuously cool down and solidify by spraying cold water mists in between each roll. The rolls move at the same speed of steel feeding, which is controlled by adjusting the cross-sectional area of nozzle via a slide-gate or stopper-rod system. When the molten steel is fully solidified beyond the metallurgical length, it is cut transversely by a torch, forming semifinished slabs. The slabs are reheated to a uniform temperature and rolled into various products, such as sheets, rails, and bars.

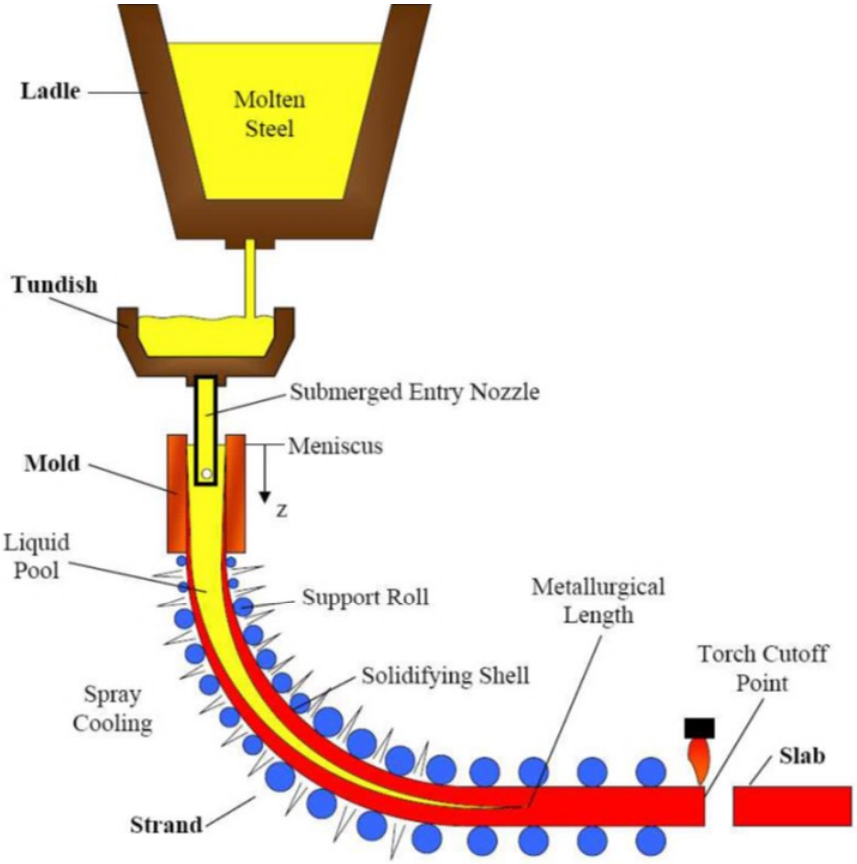


Figure 1.1 Continuous casting process schematic [3]

The process starts with a "dummy bar" plugged at the bottom of the mold. When enough metal has solidified against the "dummy bar" like in a conventional ingot casting route, the bar is slowly withdrawn along the casting direction and through the casting machine. Steady state condition then gradually develops. Once starting, the operation can last for up to a few weeks.

Figure 1.2 shows a close-up schematic of the flow inside a SEN and mold. The molten steel flows from tundish, through the SEN, into the mold cavity, hitting the narrow face and getting split into two streams. A classic double-roll flow pattern is formed. The upward stream forms a smaller upper recirculation zone by being deflected up towards the top slag layer and moving from narrow face (NF) towards SEN. In this process, slag may get entrained into the molten steel as detrimental particles. The downward stream forms a larger lower recirculation zone and transports the molten steel deeper down the strand. In this process, argon gas is usually injected through the nozzle refractory to prevent clogging [4].

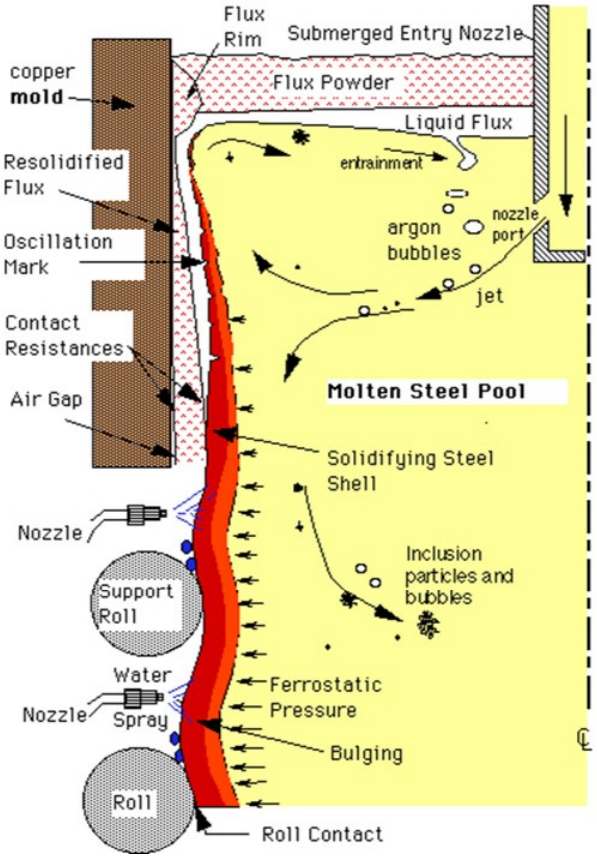


Figure 1.2 Longitudinal slab casting schematic in SEN and mold region [5]

During the transport, argon bubble may pick up inclusions, forming a inclusion-laden cluster. The argon bubbles, slag particles, inclusions, and clusters can flow with the molten steel, touch the solidification front and subsurface hooks, and be captured. If these detrimental particles remain in the product, then surface defects such as "slivers" may form during subsequent rolling operations, or they may lower the steel fatigue life by causing local internal stress concentration, thus lowering the steel quality.

1.2 Research Motivations and Objectives

With billions of metric tonnes of steel produced by continuous casting process per year, any small improvements of this process can potentially result in great cost saving and quality enhancing. Turbulent molten steel flow has significant effect on final steel quality, as it can affect the transport and capture of particles which may form detrimental defects once trapped by the solidifying shell. Steel flow pattern is greatly coupled with argon gas due to its buoyancy effect, especially under high argon injection rate. The effect of superheat (defined as the difference between the molten steel temperature and its liquidus temperature) on flow pattern in the mold and lower strand is lacking, which is worth investigating, as the

superheat dissipation is crucial for shell thickness, surface hook formation, breakout, and possibly lower-strand flow pattern where inertia is negligible.

Mold-ElectroMagnetic Stirrer (M-EMS) can wash particles away from the dendritic solidification front and lessen particle capture, especially near the meniscus, and lead to a more uniform mold surface temperature to lessen the associated problems of meniscus freezing and deep hook depth.

However, even with modern technologies, the flow-related defects in steel products are still inevitable. The harsh environment in the real-world steel plants makes it hard to fully understand how different parameters affect the flow pattern. Therefore, there is a great incentive to use numerical modeling, with different parameters, to predict and understand the mechanism and possible solution to various industrial steel defects.

This research aims to develop multiple three-dimensional computational fluid dynamics (CFD) models to gain a fundamental understanding of the relations between turbulent-multiphase flow pattern, superheat dissipation, and particle transport and capture phenomenon during the continuous casting process of steel. Specifically, the following topics are studied:

- Effect of SEN design on strand flow pattern and particle transport and capture;
- Effect of superheat on strand flow pattern and particle transport and capture;
- Effect of M-EMS on strand flow pattern and particle transport and capture;
- Effect of change in ladle exchange temperature on tundish flow pattern and inclusions entering the mold region;
- Effect of tundish insertion nozzle misalignment on tundish flow pattern and inclusion entering the mold region;

1.3 Dissertation Outline

This dissertation is divided into 10 chapters.

Chapter 2 is a literature review of previous work on the turbulent multiphase flow, superheat dissipation, particle transport and capture during continuous casting process, preceded by an introduction chapter (Chapter 1), and succeeded the results chapters (Chapters 3-9), and a conclusion chapter (Chapter 10).

Chapter 3 is modified from a 2019 AISTech paper [6], which compares the flow pattern in 2 casters with different SEN designs, using a computational fluid dynamics model system developed by the author in the commercial software ANSYS FLUENT and User Defined Functions. Chapter 3 uses the PFSG model

developed by H. Olia, a two-stage bubble-formation model developed by R. Liu, the CON1D model developed by Y. Meng and many other researchers in the Continuous Casting Center (CCC) with direction from Prof. Brian. G. Thomas. An SST k-ω turbulence model, and an Eulerian-Eulerian two-phase model in the commercial CFD code, Ansys Fluent, are used in Chapter 3. Both the casting conditions and measurements used in this chapter are provided by Dr. Xiaoming Ruan of Baosteel. Based on Ultra-sonic Test (UT) measurements, one SEN is proved to give better steel quality than the other. The simulation results suggest that the good SEN gives an upwards flow right out of the nozzle ports, which transports more large bubbles into the top slag layer, thus giving a better steel quality. The results are validated with plant measurements of nail-board dipping tests of the top-surface velocity. The UT maps show a capture band at 1/4 across the slab/plate thickness from the inner radius (IR) side. The simulation suggests that a stagnation region from a longitudinal-recirculation zone at the 1/4 longitudinal plane of the strand width turns out to be the reason for this capture band. Other co-authors contributed the following to this chapter: Dr. Seong-Mook Cho advised and discussed with the author about the geometry / mesh generation and modeling methodology; Hamed Olia (advised by Prof. Brian G. Thomas) developed the PFSG model, which is used to predict the hot steel pressure inside the SEN; Lipsa Das (advised by Prof. Brian G. Thomas) and the author developed a MATLAB code to predict the primary dendrite arm spacing down the caster, which is presented in the original paper and in Chapter 4, and later coded into the entrapment / engulfment User Defined Function (UDF) for Ansys Fluent; Prof. Brian G. Thomas advised the whole research. The author modified the mass / momentum sink UDFs for a curved strand and argon escaping into the top slag layer, performed all simulations, analyzed the results, and wrote up the original paper draft.

Chapter 4 is modified from a paper published in the Journal of Processes in July 2022. This chapter investigates the effect of argon gas injection and superheat on nozzle and mold flow pattern, based on the model system for the good SEN from Chapter 3. In addition to models used in Chapter 3, Chapter 4 uses a standard k-ϵ model, and a Discrete Phase model (DPM) in Ansys Fluent, a primary dendrite arm spacing model, developed by both Lipsa Das and the author, a heat transfer model for fluid flow in Ansys Fluent, a slag energy-balance model developed by the author, a hook capture model, developed by K. Jin, entrapment and engulfment capture models, developed by Q. Yuan, S. Mahmood, K. Jin, and others under the guidance of Prof. Thomas, and modified/improved here by the author. The casting conditions and measurements used in this chapter are provided by Dr. Xiaoming Ruan of Baosteel. Other than the models mentioned in Chapter 3, several capture models are used in Chapter 4, including a primary dendrite arm spacing (PDAS) model developed by both the author and Lipsa Das, a hook capture UDF developed by Kai Jin, an entrapment / engulfment capture UDF developed by Yuan Quan and Sana Mahmood, and a

slag energy balance model developed by the author to calculate the heat flux from the molten steel into the top slag layer. Energy sink terms in the UDFs for superheat dissipation are developed by the author. All the model developers mentioned here are advised by Prof. Brian G. Thomas. To achieve better convergence, only half of the domain is modeled, and the mesh is modified to be aligned with the jet-flow direction to achieve better convergence. Better agreement with plant nail-board measurements of surface velocity is seen. Simulation suggests that higher superheat can complicate the flow by creating multiple recirculation zones due to steel thermal-buoyancy force. Shallower capture locations and less capture fraction for large bubbles (d ¿ 0.6 mm) are seen with higher superheat. Lower superheat leads to deeper hook and 11 1 mm bubbles captured per meter of slab. There are two reasons for the capture-band formation: (1) the stagnation region near the IR strand and (2) the downwards fluid velocity balancing the particle terminal velocity. The calculated number of bubbles captured inside the IR and outer radius (OR) capture-band region matches the UT maps well.

In Chapter 5, A Large Eddy Simulation (LES) model is developed in ANSYS FLUENT by the author, to study the transient asymmetrical flow in a full-domain nozzle and strand, based on the good SEN case in Chapters 3 and 4.

Chapter 6 investigates the effect of M-EMS on nozzle and mold flow pattern in ANSYS FLUENT. A CFD model and methodology are developed by the author. Chapter 6 uses the same models as Chapter 4, except for the heat transfer model and slag energy-balance model. The casting conditions and measurements of Lorenz force distribution are provided by Dr. Xiaoming Ruan of Baosteel. The M-EMS pushes the jet downwards and leads to deeper capture of bubbles. The capture of bubbles near the meniscus is lowered by M-EMS due to higher velocity washing bubbles away from hook zone and dendrite front. Validation of the numerical model is needed. But the new method developed in this chapter can shed some light on future simulations of M-EMS and molten steel flow.

In Chapter 7, the author developed a CFD model in ANSYS FLUENT to investigate the effect of ladle exchange from higher to lower temperature on the transient tundish flow and inclusion exit fraction at the tundish outlet, using a single-phase standard-k-ϵ Reynolds-Averaged Navier-Stokes (RANS) model and DPM model in Ansys Fluent with appropriate user-defined functions. The results are compared with steady-state simulation with 30 K superheat. The casting conditions and measurements of inclusion concentration are provided by Dr. Xiaoming Ruan of Baosteel. The colder steel tends to stay at the bottom after being fed into the hotter steel, which results in a short-cut path along the tundish bottom for the inclusion to flow through and exit the tundish outlet, into the SEN and mold. This exit fractions of different inclusions are quantified with respect to casting time. The short-cut path can lead to a worse steel quality for a $\sim$13.8-m-long slab.

Chapter 8 details how the misalignment of tundish insertion nozzle affects the flow pattern and inclusion removal, with the same casting conditions and geometries in Chapter 7, using a single-phase SST-k-ω model and DPM model in Ansys Fluent. The asymmetrical geometry leads to an asymmetrical flow and more large inclusions exiting the tundish outlet on one side than the other. Chapter 8 also studies the effect of different superheat on tundish flow pattern and inclusion removal. The results show that 10 K and 30 K superheat has negligible effect on tundish flow pattern due to the similar temperature gradient, thus it won't affect the inclusion removal too much.

Chapter 9 details an ongoing work on the engulfment capture UDF, based on the flow pattern from the lower superheat case in Chapter 4. the results are compared with the old-version UDF. The new UDF gives less capture of large bubbles ($>$ 0.3mm), especially near the jet region where the high-speed jet flow washes the particles away. However, the new UDF gives no large bubbles (0.6 mm diameter) captured inside the capture-band region as opposed to the old UDF (Chapter 4), possibly due to the neglect of transient turbulence in the simulation.

Chapter 10 provides a summary of results, conclusions, and potential future work.

Appendix A shows the UDFs used in this thesis, and finally, Appendix B shows the permission of re-print of the conference and journal papers from the organizations and co-authors.

CHAPTER 2
LITERATURE REVIEW

Much research has been done on tundish and mold flow both experimentally and numerically, using direct measurements and computational models.

2.1 SEN Design

Submerged entry nozzle (SEN) design is crucial for flow pattern in the mold and particle capture. If the nozzle jet penetrates downwards, it tends to transport more particles down the strand and form more defects; if the jet angle is flat or the nozzle's submerged depth is too shallow, the jet flow will interact with the top slag layer or transport too much superheat to the thin shell, causing more slag entrainment and possible break out, respectively; If the jet angle is pointing too deep, or the submerged depth is too deep, then the critical meniscus region receives too little superheat, so the meniscus can solidify to form hooks [7]. Much research has been done on the effect of nozzle design or nozzle clogging on flow pattern. [8–11]. A combination of water model and numerical simulation has been proved to be an effective method to determine the optimal nozzle geometry for better flow pattern, temperature distribution, and shell thickness [8–10]. Nozzle clogging changes nozzle internal geometry in a way that more inclusions enter the non-clogged side, which leads to lower quality on that side. Clogging also causes asymmetrical flow pattern and temperature distribution, increasing the risk of breakouts [11]. Thus, there is a great incentive to compare different SEN designs, for better flow pattern and steel quality. This study compares two plant slide-gate-controlled SENs in Chapter 4: one gives lower slab rejection rate (SEN 1) than the other one (SEN 2), based on real-world UT measurements.

2.2 Superheat in Strand Flow

Molten steel superheat should have significant effect on bubble capture as well, as they can change the flow pattern in the strand region by thermal buoyancy. Some research have been done to investigate superheat dissipation and transport during continuous casting of steel [12, 13]. These models show that most of the superheat is removed in the mold region, or just below [12, 13]. The temperature decreases continuously with distance along the path travelled by the flowing jet, with the lowest temperatures found at the meniscus region, near the narrow faces and SEN, for a typical double-roll flow pattern[12, 13]. However, studies showing how different superheat affects the fluid flow and particle capture are lacking. This study investigates the effect of superheat on flow pattern in the liquid pool of a 7-m-long vertical and bending caster, and on bubble capture fraction and locations in Chapter 5.

2.3 Particle Capture

In continuous casting of steel, fluid flow in the nozzle and strand can potentially entrap/engulf detrimental particles, such as inclusions from upstream ladle refining process, tundish flow and mold top-slag layer, and argon bubbles. Argon gas is often injected through porous refractory material in the nozzle in order to lessen nozzle clogging [6].

Although most of the gas escapes through the top-surface slag layer, some can become (1) captured by subsurface hooks near top surface [14, 15], (2) flowing in between primary dendrite arms and entrapped by solidifying steel, or (3) engulfed by growing dendrites if the bubble diameter is bigger than primary dendrite arm spacing (PADS) and the forces on it balance out, causing permanent surface and internal defects in the final product, such as pinholes, slivers, blisters and expensive rejects [16]. Thus, there is a great incentive to understand the complex phenomena of bubble transport and capture by fluid flow and solidification, to optimize the casting conditions, nozzle geometries, and steel quality.

Many researchers have investigated the argon bubble and non-metallic inclusion transport and capture phenomenon by simple capture criterion which assumes instant capture when a bubble or inclusion touches solidification front [17–21]. Although this method is valid for particles smaller than PDAS, it naturally overpredicts the capture of particles larger than PDAS, because the engulfment phenomenon, due to particle's force balance and stationary rest on dendritic solidification front, is not taken into account. Efforts has been made to calculate the capture of large bubbles. Liu et al. [21] assumed that bubbles with diameter larger than 500 μm are not captured by the solidification front to compensate for the overprediction from simple criterion, but this method underpredicts the capture fraction. Chen et al. [22] assumed that inclusions are captured by the solidification front where the liquid fraction and the fluid flow speed are less than 0.6 and 0.07 m/s, respectively. Good agreement is seen but this method can only be used when solidifying mushy zone is modeled at the same time, not for a simple multi-phase fluid flow simulation.

Thomas et al. [18, 19, 23] developed an entrapment / engulfment capture criterion for particles like inclusions and argon bubbles, which can decide if a larger-than-PDAS particle should be captured or not, based on local force balance at solidification front. Kai et al. [24] compared the simple criterion and the entrapment / engulfment capture criterion on argon bubble capture, and the latter one gave a good validation with plant step-milling measurements. Two recent studies have used the entrapment / engulfment criterion for both argon bubble and inclusion on both straight and bending section of two plant casters [6, 25]. Simulation results show a capture band of bubbles and inclusions at around 1/4 and 3/4 across the slab thickness from the IR side of the caster [6, 22, 25, 26], which agrees with the UT maps. Cho

et al. [25] shows a good agreement between the UT maps and capture rate of large inclusions (200-300 μm) predicted by the entrapment / engulfment capture criterion.

Previous research with this entrapment / engulfment criterion underpredicts the bubbles captured near the slab surface, possibly due to the model neglecting meniscus hook capture [24, 27]. Subsurface hook formation during initial solidification of continuous casting can degrade the steel quality, by entrapping argon bubbles and non-metallic inclusions while moving down, forming near-surface defects. Many previous work investigated the effect of operating parameters on subsurface hooks [28–31]. G. Lee et al. developed an empirical equation to estimate the hook depth for ultra-low carbon steel [32]. Increasing superheat delivery to the meniscus region with increasing casting speed has a strong effect on decreasing hook depths [32]. Recently, a hook capture mechanism was developed, by considering a bubble as captured if it stays long enough in a hook zone [33]. Therefore, both the advanced capture criterion and the hook capture mechanism are used in this study to predict the argon-bubble capture fraction and location in continuous casting process.

2.4 Mold-Electromagnetic Stirring

One way to lessen the particle capture is to use a moving magnetic field to wash the particles away from the dendrite tips. This is achieved by using Alternating Current (AC) through a set of magnets placed in the upper mold region, with increasing phase shift, to achieve apparent motion of the magnetic field near each of the two wide faces (WF) of the mold [34]. Four sets of magnets are installed, two on each wide face, and can generate three moving fields to slow the jet, accelerate the jet, or rotate the mold flow horizontally. The last moving field is called Electromagnetic-Stirring in the Mold (M-EMS) [35, 36], which moves the molten steel in opposite direction on each wide face, thus horizontally rotating the mold flow around its perimeter. M-EMS can wash particles away from the dendritic solidification front and lessen particle capture especially near the meniscus, and lead to more uniform mold surface temperature to lessen the associated problems of meniscus freezing and deep hook depth.

2.5 Molten-Steel-Argon-Gas Multiphase Turbulent Model

To study the molten-steel-argon-gas two-phase fluid flow in nozzle and mold with different casting conditions, and its subsequent impact on particle capture, many researches have been conducted using three-dimensional CFD model[6, 21, 22, 24, 26, 27, 37–41], with both Eulerian-Eulerian[6, 13, 38–40] and Eulerian-Lagrangine[6, 21, 22, 24, 26, 27, 37] approach. Increasing argon flow rate can change the mold flow from a classic double-roll to a single-roll flow pattern with surface flow moving away from SEN towards narrow face [39, 40, 42]. Excessive gas fractions have been observed to lead to oscillating and

asymmetric flow [43, 44]. Therefore, it is important to have a two-way-coupled two-phase flow to consider the effect of argon gas on molten steel, especially under high gas fraction condition. Kai et al. developed a Reynolds-Averaged Navier– Stokes (RANS) k-ϵ model, coupled with Discrete Phase Model (DPM) and advanced capture criterion, to investigate argon bubble capture and location in a continuous casting nozzle and straight mold [45]. The turbulent dispersion of each bubble can be mimicked by Random Walk Model (RWM) [19–21, 23, 37, 45–50]. Results show that 85 pct of small (<0.08 mm) bubbles are captured. A very small fraction of large bubbles is captured (<0.02 pct) [45].Kai et al. also compared RANS with Large-Eddy-Simulation (LES) turbulence model, coupled with DPM[27]. The LES model predicts that a smaller number of bubbles are captured in comparison with the RANS approach and the random walk model, especially for $0.2 \leq \text{dp} \leq 0.3$ mm bubbles [27]. The capture fraction is observed to be only 1 percent which is much less than that predicted by previous RANS simulation (60 pct) [27]. The causes for this discrepancy may be the differences in the flow pattern predicted by the two models and the use of random walk in the RANS model which over-predicts the probability of the bubbles hitting the walls [27].

However, the RANS-Lagrangian-coupled method is expensive, as it needs to solve the trajectory of each bubble in a transient way. Moreover, transient behavior is less accurate with RANS [51]. LES can more accurately resolve the details of transient flow, but the LES–Lagrangian-coupled method needs even more computational time, because it needs fine mesh and small time step size to solve the small-scale eddies.

2.6 Tundish Flow

Tundish, served as a buffer zone and regulator of steel flow between the ladle and continuous casting molds, plays a vital role in steel cleanliness by removing the inclusions from upstream ladle reoxidation process into its top slag layer. During this process, the hotter or colder molten steel will be poured from ladle into tundish. The temperature of the inlet stream from the ladle may vary from different heats, or on the ladle's teeming time [52]. The thermal buoyancy effect induced by temperature difference between liquid steel may change the flow pattern inside tundish significantly and affects the flotation and removal of inclusions.

Therefore, there is a great incentive to study this transient behavior during ladle exchange, especially from exchange from hot steam to cold stream because the colder molten steel tends to move downwards when feeding into hotter steel due to its higher density, thus changing the flow pattern and inclusion removal drastically.

Extensive research has been done to investigate the non-isothermal fluid flow in tundish with numerical modeling and water experiments, but with a focus mainly on optimization of flow control device and its effect on inclusion removal [50, 53–59]. Joo et al. (1993) [53] proposed that even a small temperature drop

($1°C$) would make natural convection currents significant in a 1.1-m-deep tundish, particularly at the end of tundish walls, owing to small flow velocities and large heat losses in these regions. They also concluded that small inclusions ($\leq 40\mu$m) are less readily removed into slag layer due to its small stoke rising velocity and tendency to move with the flow.

Far few studies are conducted on the effect of inlet-stream temperature rise or drop on the effect of tundish flow pattern and inclusion removal. Chakraborty and Sahai (1991)[60] developed a transient, three-dimensional model to study the effect of varying ladle temperature on the fluid flow and heat transfer phenomena in a typical twin strand slab caster tundish. With an incoming steel temperature declining at a rate of $0.5°C/min$ over a casting period of 50min, the cooler incoming steel starts to flow along the bottom of the tundish instead of the normal top-free-surface-directed flow. Thus, the inclusion flotation and removal are expected to be worse. Q. Tian et al. (2012)[61] investigated the effect of both $0.5°C/min$ and $0.25°C/min$ inlet cooling rate on the fluid flow and heat transfer in a flexible thin slab casting tundish. With $0.5°C/min$ cooling rate, the horizontal flow along the top surface turns to flow along the bottom of tundish, which shortens the fluid flow routes and deteriorates the removal of non-metallic inclusions from molten steel; With $0.25°C/min$ cooling rate, the horizontal flow sustains through the whole casting process. However, both studies did not quantify the effect of fluid flow on inclusion removal.

Chatterjee and Chattopadhyay (2016)[62] used both water experiment and numerical modeling to study the effect of non-isothermal conditions on fluid flow in a multi-strand billet caster tundish. It was observed that using inlet temperature step down by $10°C$ or up by $23°C$ for water and steel, respectively, changed the fluid patterns significantly, especially in regions far away from the inlet due to less inertia effect [62]. The incoming hotter fluid in case of step-up condition resulted in an upward buoyant flow which aided flotation of inclusions, whereas the incoming colder fluid in case of step-down condition dragged the inclusions into the SENs, proving to be disastrous regarding steel quality [62]. The inclusion trajectories and removal are quantified during the inlet temperature change. However, this study is conducted on a tundish without any internal furniture to optimize its flow and inclusion removal, and the maximum inclusion size simulated is only 150μm.

The molten steel is poured into the tundish from upstream ladle furnace, through an insertion nozzle (ladle shroud). An impact box is usually located at the center of the tundish bottom to inhibit the turbulence and deflect the flow upwards to remove particles. Multiple-strand casters are usually connected to the tundish to improve output, and any misalignment between the insertion nozzle and the impact box may lead to asymmetrical flow in the tundish. The asymmetrical flow can transport more particles to one side of the tundish than another, leading to much worse steel quality on that side. Literature on misalignment of tundish insertion nozzle is lacking. This thesis investigates this phenomenon in Chapter 9.

CHAPTER 3

MODELING OF MULTIPHASE FLOW WITH TWO DIFFERENT SENS

Modified from a paper published in The AISTech 2019 Conference [1].

Mingyi Liang[2,3], Seong-Mook Cho[4], Hamed Olia[2], Lipsa Das [2], Xiaoming Ruan [5], Brian G. Thomas [6 7]

3.1 Abstract

In continuous casting, argon gas is injected to prevent nozzle clogging, which can affect the flow pattern and may be captured into the steel shell resulting in internal and surface defects in final steel products. SEN design can affect the flow pattern and the flow-related particle defects. Three-dimensional multiphase-flow models are applied to quantify the flow pattern in a continuous slab caster for two different SEN designs. The computational model is validated with plant measurements of surface velocity.

3.2 Introduction

In continuous casting of steel, fluid flow in the nozzle and strand can potentially entrap detrimental particles, including inclusions from upstream refining operations, mold slag and argon bubbles. Argon gas is often injected through porous regions in the nozzle refractory in order to lessen nozzle clogging problems [63]. The gas bubbles often become coated with inclusion particles they contact. Although most of the gas escapes through the top-surface slag layer, some can become entrapped into either solidifying steel or hooks near the meniscus, causing permanent defects in the final product, such as pinholes, slivers, blisters and expensive rejects [16].

Thus, there is great incentive to understand particle transport and capture phenomena and to optimize the flow pattern in continuous steel casting to minimize particle entrapment and the associated quality problems. However, due to the harsh environment of the process, it is impossible to observe the internal flow behavior and particle defect formation in the commercial process while they occur. Particle capture is also difficult to simulate in water models or lab-scale models. Thus, combining computational models of all the complex phenomena together with feasible plant measurements, such as surface flow velocity and the locations of defects in the as-cast slabs, is a useful way to investigate this problem.

---

[1] Reprinted with permission of AIStech 2019 Proceedings, volume 3, 2219-2231, and of the co-authors: Seong-Mook Cho, Hamed Olia, Lipsa Das, and Xiaoming Ruan

[2] Graduate student at the Colorado School of Mines
[3] Primary researcher and author
[4] Professor, Department of Metallurgical Engineering, Pukyong National University
[5] Steelmaking Research Department, Research Institute Baoshan Iron & Steel Co., Ltd.
[6] Author for correspondence
[7] Professor, Mechanical Engineering Department, Colorado School of Mines

In this work, a new, computationally-efficient modeling methodology has been developed and validated with plant measurements to quantify multiphase flow of molten steel and argon during steady continuous casting of steel slabs. Two cases shown in Table 3.1 have been simulated on the same caster, for conditions which tend to produce higher quality steel (Case 1) and increased frequency of internal particle-capture defects (Case 2). Case 1 has shallower submergence and less argon. In addition, Case 1 has a downward SEN, and SEN in Case 2 has horizontal ports. The flow patterns in nozzle and strand were compared to investigate internal particle defects.

Table 3.1 Caster dimensions and process conditions

Parameters	Symbols	Case 1	Case 2
Port down angle	θ	15°	0°
Slab thickness/width	t_s/w_s	300 mm/2300 mm	
Tundish height	h_t	1400 mm	
Casting speed	v_c	0.6 m/min	
Steel flow rate	m	2.9 tonne/min	
Slide gate opening	E/E_{max}	62%	
Submergence depth	d_s	140mm	180mm
Argon flow rate			
via UTN: Hot/Cold	Q_{h_utn}/Q_{s_utn}	23.3LPM/5.2SLPM	23.2LPM/5.4SLPM
via upper plate: Hot/Cold	Q_{h_plate}/Q_{s_plate}	9.1LPM/2.2SLPM	16.2LPM/3.9SLPM
Argon volume fraction	f_{Ar}	7.2%	8.7%
Molten steel density	ρ_s	7000kg/m^3	
Molten steel dynamic viscosity	μ_s	0.0063kg/m·s	
Ar viscosity (kg/m·s)	μ_g	2.125×10^{-5}	

3.3 Computational Models

Various computational models are used in this chapter.

3.3.1 PFSG and Two-stage Mean Bubble Size Models

The PFSG model is a one-dimensional pressure energy model to calculate the pressure distribution and flow rate in slide-gate nozzle systems for the argon-molten-steel flow system including argon gas expansion by solving a system of Bernoulli equation [64]. This model has been verified with CFD simulations and validated with plant measurements and water model measurements, with errors of less than 6% [64]. With the hot molten steel pressure P_h at the injection location predicted by the PFSG model, the hot argon gas flow rate Q_h can be calculated by:

$$Q_h = \frac{T_h P_s}{T_s P_h} Q_s \tag{3.1}$$

where T_h is the casting temperature in K, T_s is the standard condition temperature, 298.15 K, P_s is the standard condition pressure, 1 atm.

With Q_h, a two-stage mean bubble size model can be used to predict the mean argon bubble diameter $d_{p,mean}$ [65]. During the first stage, argon bubbles expand while holding onto the nozzle refractory pores. In the second stage, argon bubbles get detached from the nozzle wall and move with the transverse liquid stream. The final bubble size $d_{p,mean}$ (at the time of detaching from the pore) is derived from certain detaching criteria based on experimental observations, and is used in the steady-state Eulerian-Eulerian model.

3.3.2 Heat Transfer and Solidification: CON1D

Shell thickness profiles in the mold and strand on both wide and narrow faces are calculated from the CON1D model [66]. Then the strand shell thickness can be roughly characterized by a solidification constant, based on fitting the CON1D output to:

$$s = K\sqrt{time} = K\sqrt{\frac{l_c}{v_c}} \tag{3.2}$$

where l_c is the distance below meniscus along the strand (m) and v_c is the casting speed (m/s). In addition to defining the shape of the liquid pool for the model domain, the shell thickness profile is implemented into the mass and momentum sink UDF.

The solidification constant K is 3 mm/$\sqrt{s}$, and the solidification-shell thickness revolution with respect to the distance below meniscus along the strand on both NF and WF are shown in Figure 3.1.

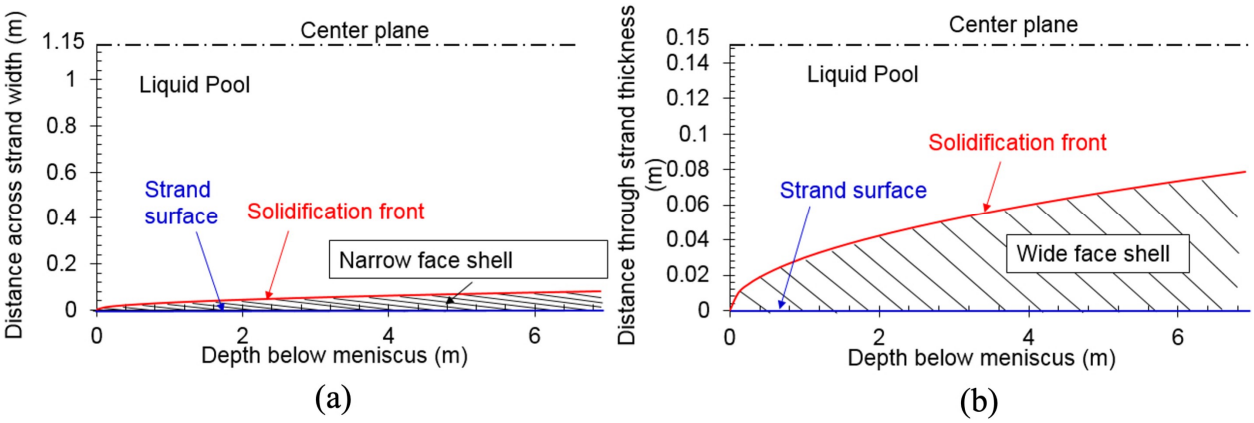


Figure 3.1 Steel shell thickness profiles down the steel strand (a) Narrow face, and (b) Wide face

The shell thickness is then used to build the liquid pool of the strand. In this way, the modeling of solidification and the extra fine mesh near the solidification front are avoided. The domain is constant

throughout the simulation.

3.3.3 Governing Equations for Multiphase Flow

The iso-thermal steady-state Eulerian-Eulerian multiphase flow model with SST k-ω turbulence model are adopted for the molten-steel-argon-gas two-phase flow simulation. Mass and momentum sink UDFs are applied at the wall-adjacent cells. The UDFs are listed in Appendix A.

A three-dimensional steady-state Eulerian-Eulerian model is employed to simulate the multiphase molten-steel-argon-gas flow in the liquid pool of the caster, using commercial finite-volume software ANSYS FLUENT [67]. This model is considered as more accurate than other continuous-continuous phase models (mixture and VOF (volume of fluid)) as it solves separate equations for each phase. User-Defined Functions (UDFs) are used to account for mass and momentum sinks around the liquid pool boundary due to steel solidification [18, 68]. The governing equations of Eulerian-Eulerian model solve separate continuity, Navier-Stokes and energy equations for each phase as follows:

$$\frac{\partial}{\partial x_i}\alpha_q\rho_q u_{iq} = S_{q-mass-sink} \tag{3.3}$$

$$\frac{\partial \alpha_s\rho_s u_j u_{is}}{\partial x_j} = -\alpha_s\frac{\partial P_2}{\partial x_i} + \frac{\partial}{\partial x_j}[(\mu_s+\mu_{ts})(\frac{\partial u_{is}}{\partial x_j}+\frac{\partial u_{js}}{\partial x_i})] + \alpha_s F_{Bi,g} - R_{igs} + \sum F_{is} + F_{Bi,t} + S_{is-mom-sink} \tag{3.4}$$

$$\frac{\partial \alpha_g\rho_g u_j u_{ig}}{\partial x_j} = -\alpha_g\frac{\partial P_2}{\partial x_i} + \frac{\partial}{\partial x_j}[(\mu_g+\mu_{tg})(\frac{\partial u_{ig}}{\partial x_j}+\frac{\partial u_{jg}}{\partial x_i})] + \alpha_g F_{Bi,g} - R_{igs} + \sum F_{ig} + S_{ig-mom-sink} \tag{3.5}$$

$$F_{is} = -F_{ig} = F_{iVM} + F_{iD} = \overbrace{0.5\alpha_g\rho_s(\frac{Du_{is}}{Dt}-\frac{Du_{ig}}{Dt})}^{\text{virtual mass force, } F_{iVM}} - \underbrace{\frac{3}{4}(\frac{C_D}{d_p})\rho_s\alpha_g \mid u_{ig}-u_{is} \mid (u_{ig}-u_{is})}_{\text{drag force, } F_{iD}} \tag{3.6}$$

where subscript q is either s or g denoting steel and argon gas phase. The sum of each phase's volume fraction is 1 in each cell (α_s+α_g=1). u_{is} and u_{ig} are steel and argon gas velocity tensor. P_2 is the pressure shared by all phases and g_i is gravitational acceleration (9.81 m/s^2 in -y direction). μ_s and μ_g are the dynamic viscosity of molten steel and argon gas; the turbulent viscosity μ_{ts} and μ_{tg} are calculated by the SST k-ω model.

Steel mass and momentum sinks are applied to the cells adjacent to the solidification front (narrow face (NF) and wide face (WF) walls) to account for mass and momentum loss due to liquid phase transitioning into solid steel. The steel mass sink term is defined as follows:

$$S_{s-mass-sink} = \rho_s v_c A_c V_{cell}^{-1} \tag{3.7}$$

the projected area of wall-adjacent cell face in the casting direction, A_c, can be calculated by equation below:

$$A_c = A_f \times sin\theta_c \tag{3.8}$$

where A_f is the wall-adjacent face area in m^2, and θ_c is the angle between the face and casting direction. Previous work has developed UDFs to calculate θ_c on the straight part of the strand [18, 19, 24, 27, 33, 68], but a new method is needed to find θ_c for the curved part of strand. This study calculates θ_c by taking derivative of the shell-thickness empirical equation and take arcsine of the results:

$$\theta_c = arcsin(\frac{\partial s}{\partial l_c}) = arcsin(0.5K\sqrt{\frac{v_c}{l_c}}) \tag{3.9}$$

Argon gas sink terms are only applied to top surface (wall) to simulate the process of argon gas escaping into the top slag layer, because the bubble diameter in the Eulerian-Eulerian model is set as the calculated mean bubble diameter, which is large and means that bubbles tend to move into slag layer immediately after they flow through the nozzle port. For argon gas phase, the mass sink term is as follows:

$$S_{g-mass-sink} = \rho_g v_{gy} A_y \alpha_g V_{cell}^{-1} \tag{3.10}$$

where v_{gy} is gas-phase vertical velocity towards the top surface (in +y direction, m/s), A_y is the top-wall-adjacent cell face area (m^2), and α_g is the gas volume fraction in the top-wall-adjacent cell.

Momentum sinks can be calculated based on mass sink term:

$$S_{iq-momentum-sink} = S_{q-mass-sink} u_{iq} \tag{3.11}$$

R_{igs} is the source term for interphase force and is defined as:

$$R_{igs} = \alpha_g(1-\alpha_g)\rho_g[\overbrace{\frac{C_{D1}Re_p}{24}}^{f_{drag}}](u_{ig}-u_{is})\overbrace{[\frac{\rho_g d_p^2}{18\mu_s}]^{-1}}^{\tau} \tag{3.12}$$

where τ is particle relaxation time which represents a time scale for a particle to respond to changes in the surrounding flow. d_p is Ar bubble diameter. C_{D1} is drag coefficient (defined by Equation 3.14), and Re_p is particle Reynolds number due to the relative difference between the particle and local fluid velocity, which is defined by

$$Re_p = \frac{\rho_s \mid u_{ig} - u_{is} \mid d_p}{\mu_s} \tag{3.13}$$

In Equation 3.4 and 3.5, $\sum F_{is}$ and $\sum F_{ig}$ are additional forces on the liquid and gas phase; F_{iVM} is virtual mass force and F_{iD} is drag force. Virtual mass force is an additional force required to accelerate the surrounding fluid when the bubble is accelerated. The gas buoyancy force, $F_{Bi,g}$, caused by the density difference between the liquid and gas phase, is $(\rho_s - \rho_g)g_i$, where g_i is the gravitational acceleration (-9.81 m/s^2 in y direction). The thermal buoyancy force, $F_{Bi,t}$, is $(\rho_s(T) - \rho_0)g_i$, where $\rho_s(T)$ is the temperature-dependent density of the steel at a given location relative to the reference density. For the drag force in Eulerian-Eulerian model, the Tomiyama drag model[69] is chosen. This model can be applied to a wide range of bubble shape regimes by including the Eotvos number Eo in the drag coefficient calculation C_{D1}.

$$C_{D1} = max(min(\frac{24}{Re_p}(1 + 0.15Re_p^{0.687}), \frac{72}{Re_p}), \frac{8}{3}\frac{Eo}{Eo+4}) \tag{3.14}$$

$$Eo = \frac{g(\rho_s - \rho_g)d_{eq}^2}{\sigma} \tag{3.15}$$

where σ is the surface tension (1.157 N/m) and d_{eq} is the sphere volume-equivalent bubble diameter. The Eotvos number Eo is the ratio of buoyancy to surface tension force.

The SST k-ω model is a hybrid of standard k-ϵ and standard k-ω model by combining them with a blending function, which gives a more accurate prediction for near-wall flow than the traditional standard k-ϵ model. The two-equation model attempts to predict turbulence by solving equations for two variables, turbulence kinetic energy (k) and specific dissipation rate (ω). The model is given by

$$\frac{\partial \rho_m k}{\partial t} + \frac{\partial \rho_m u_{im} k}{\partial x_i} = \frac{\partial}{\partial x_i}[(\mu_m + \sigma_{k1}\mu_{tm})\frac{\partial k}{\partial x_i}] - \beta^* \rho_m k\omega + \tilde{P}_k \tag{3.16}$$

$$\frac{\partial \rho_m \omega}{\partial t} + \frac{\partial \rho_m u_{im} \omega}{\partial x_i} = \frac{\partial}{\partial x_i}[(\mu_m + \sigma_\omega \mu_{tm})\frac{\partial \omega}{\partial x_i}] + \alpha_\omega \rho_m S^2 - \beta \rho_m \omega^2 + 2(1 - F_1)\rho_m \sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i} \tag{3.17}$$

where the mixture density, ρ_m, mixture turbulent viscosity, μ_m, and mixture velocity, u_{im}, are computed from $\rho_m = \alpha_g \rho_g + \alpha_s \rho_s$, $\mu_m = \alpha_g \mu_g + \alpha_s \mu_s$, and $u_{im} = \frac{\alpha_g \rho_g u_{ig} + \alpha_s \rho_s u_{is}}{\alpha_g \rho_g + \alpha_s \rho_s}$.

F_1 is a blending function. It is calculated by

$$F_1 = tanh((min[max[\frac{\sqrt{k}}{\beta^* \omega y}, \frac{500v}{y^2 \omega}], \frac{4\rho_m \sigma_{\omega 2} k}{CD_{k\omega} y^2}])^4) \tag{3.18}$$

$$CD_{k\omega} = max[2\rho_m \sigma_{\omega 2} \frac{1}{\omega} \frac{\partial k}{\partial x_i} \frac{\partial \omega}{\partial x_i}, 10^{-10}] \tag{3.19}$$

When F_1 is 0, the model becomes the standard k-ϵ model far from the wall; when F_1 is 1, the model becomes the standard k-ω model near the wall, in order to more accurately calculate the boundary layer flow.

$\tilde{P}_k$ is production limiter and is defined as

$$\tilde{P}_k = min[\mu_{tm} \frac{\partial u_i}{\partial x_j} (\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}), 10\beta^* \rho_m k\omega] \tag{3.20}$$

The mixture turbulent viscosity μ_{tm} can then be calculated by:

$$\mu_{tm} = \frac{\rho_m \alpha_1 k}{max[\alpha_1 \omega, SF_2]} \tag{3.21}$$

where F_2 is the second blending function; S is the strain rate magnitude and y is the minimum distance from the nearest wall. The turbulent viscosity μ_{ts} and μ_{tg} in Equation 3.4 and 3.5 can be calculated from $\mu_{tg} = \frac{\rho_g}{\rho_m} \mu_{tm}$ and $\mu_{ts} = \frac{\rho_s}{\rho_m} \mu_{tm}$.

Constants such as β, σ_k and σ_ω are calculated by:

$$\phi = \phi_1 F_1 + \phi_2 (1 - F_1) \tag{3.22}$$

where ϕ is one of β, σ_k and σ_ω. Other constants have values as follows: $\beta^* = 0.09$, $\alpha_1 = \frac{5}{9}$, $\alpha_\omega = 0.44$, $\beta_1 = \frac{3}{40}$, $\beta_2 = 0.0828$.

3.4 Flow Model Domain, Mesh, and Boundary Conditions

The flow model domain, including the complete slide-gate opening, nozzle, and liquid pool in the strand, is shown in Figure 3.2. The caster has a 2.665m vertical upper section, followed by bending. To take the curvature part of strand into consideration, a 7 m-long region of the strand is modeled. The width and thickness of the liquid pool domain are generated according to the solidifying steel shell thickness profile in Figure 3.1.

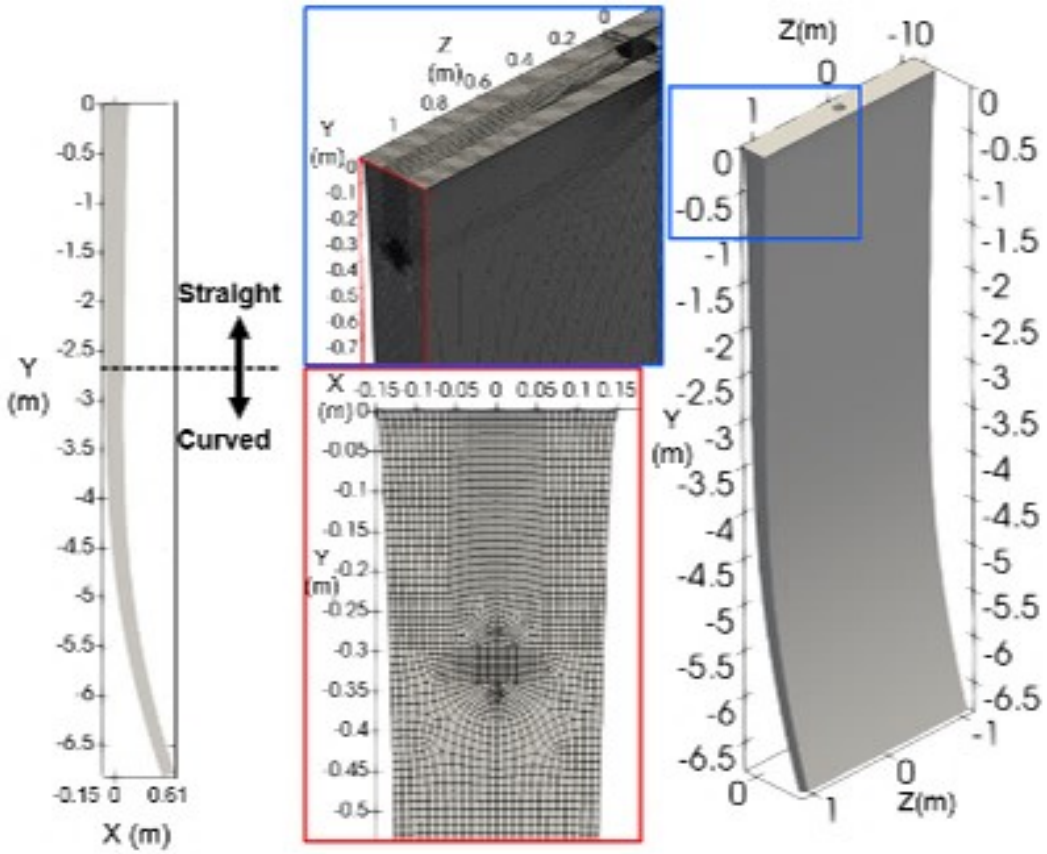


Figure 3.2 Flow domain and mesh for (a) Case 1 nozzle, (b) Case 2 nozzle, and (c) liquid pool

The mesh is created in ANSYS Workbench and includes about 0.1 million structured hexahedral cells for each nozzle domain and 4 million hexahedral cells for the liquid pool in the strand. The cell volume various from $1.07 \times 10^{-}8$ to $1.73 \times 10^{-}6 m^3$ throughout the domain, which corresponds to 2.2 - 12 mm cell length. The mesh is constructed finer near the wall and coarser in the bulk flow region, and the minimum wall Y^+ value reaches 1.2.

Boundary conditions for nozzle flow (flow-model step 1) and mold flow (flow-model step 2) simulations are given in Figure 3.3 (see page 22). For the nozzle flow simulation, constant velocity (steel volume flow rate divided by inlet area, 0.65m/s) and turbulence ($k = 10^{-5} m^2/s^2$ and $\omega = 1/s$) are given at the molten steel inlet. Argon gas volume fraction and mean bubble size are chosen for argon gas injection, through both the UTN and upper plate, at the locations shown in Figure 3.3 (a). The argon gas is injected at 0.015m/s through UTN and 0.016m/s through upper plate for case 1 and is injected at 0.016m/s (UTN) and 0.029m/s(upper plate) for case 2, based on hot argon volume flow rate divided by the inlet area. Nozzle wall has 0.001m roughness and has non-slip boundary condition. After obtaining the flow pattern in the nozzle, the outlet velocities, turbulent kinetic energy and turbulent specific dissipation rate are

interpolated onto the nozzle port outlet planes, inlets 1 and 2 in Figure 3.3 (b). The top surface is approximated as no-slip for the molten steel flow, due to the large viscosity of the slag layer. Other surfaces (wide and narrow faces) are also non-slip. The wide and narrow faces are moving at casting speed in the casting direction. A pressure outlet condition is chosen over the nozzle outlet and the strand domain bottom, according to the ferrostatic pressure head of molten steel ($\rho_s gh$), where h is the distance below the meniscus. The outlet pressure is calculated by a UDF in Appendix A. The turbulent kinetic energy and its specific dissipation rate are set as $k = 10^{-5} m^2/s^2$ and $\omega = 1/s$ for reversed flow. This methodology has been proved to be as accurate as combining the nozzle and strand as a single domain [46], and it gives better convergence and allows an upfront analysis for nozzle flow. The solutions are obtained when all the residuals reached 10^{-4}.

3.5 Nozzle Flow Results

Figure 3.4 (see page 22) shows molten steel-argon gas flow patterns with colors of flow velocity magnitude in the nozzles. Big recirculation zones with low flow velocity are observed just below the middle plate of the slide gate. This region has high gas fraction, which can cause gas pockets and/or bubble coalescence due to stagnancy of the flow. Comparing Cases 1 and 2, injecting 20% more argon in Case 2 accelerates the flow due to the smaller effective area of the bore for the steel to flow. This also elongates the recirculation zone below the slide gate.

Flow through the slide gate opening causes a clockwise swirl through the ports. As shown in Figure 3.5 (see page 23), argon flows out mainly from the center region of the nozzle port, as the swirling flow throws the molten steel to the outsides of the nozzle. Positive U_z means that the flow exits port into mold, and the white line shows where U_z is 0m/s. Backflow enters the upper 1/3 of the nozzle port. These small recirculation zones can make bubbles stay there longer, resulting in bubble coalescence [70]. Case 1 also has some argon exiting from the top of the port, which makes it easier for bubbles to escape to the slag layer, especially the large bubbles which could otherwise be captured as defects.

The jet characteristics, including jet velocity and angle, and turbulence, are calculated at the nozzle port outlet planes for both cases and are given in Table 3.2 (see page 23). The jet is defined only where there is positive outflow from the nozzle, and details are given elsewhere [71]. Case 1 has higher jet speed due to its smaller port size and greater downward jet angle due to its downward port angle.

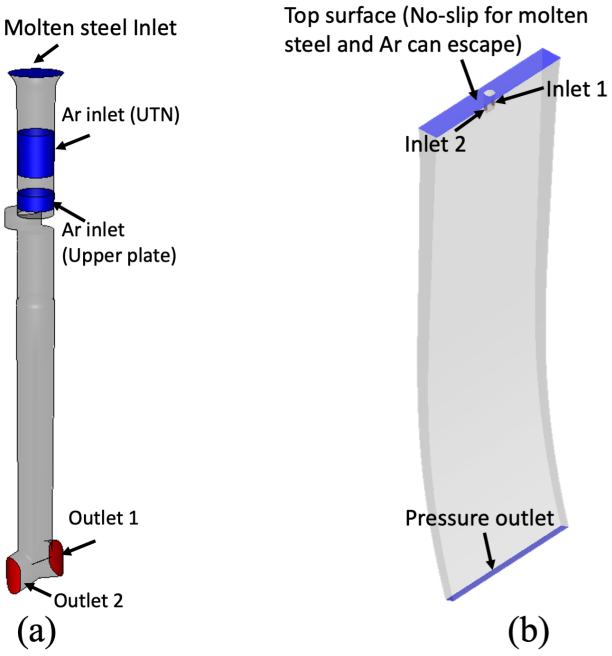


Figure 3.3 Boundary conditions for (a) step1: nozzle flow and (b) step2: mold flow

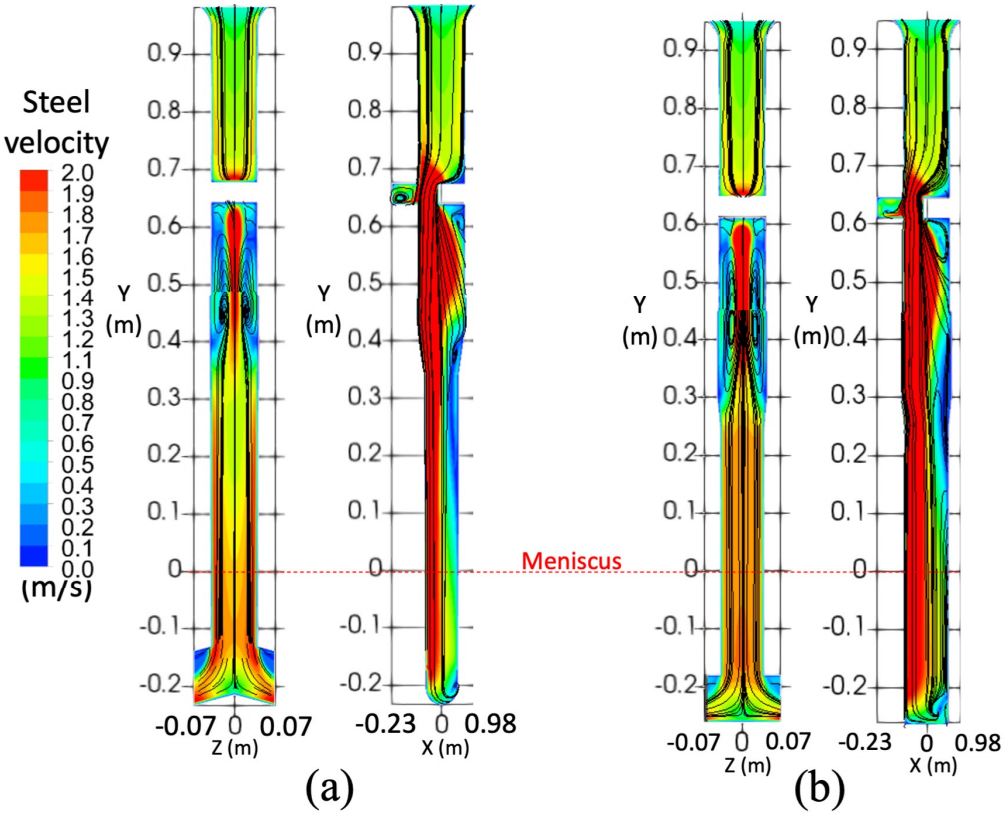


Figure 3.4 Steel velocity in the nozzle for (a) Case 1 and (b) Case 2

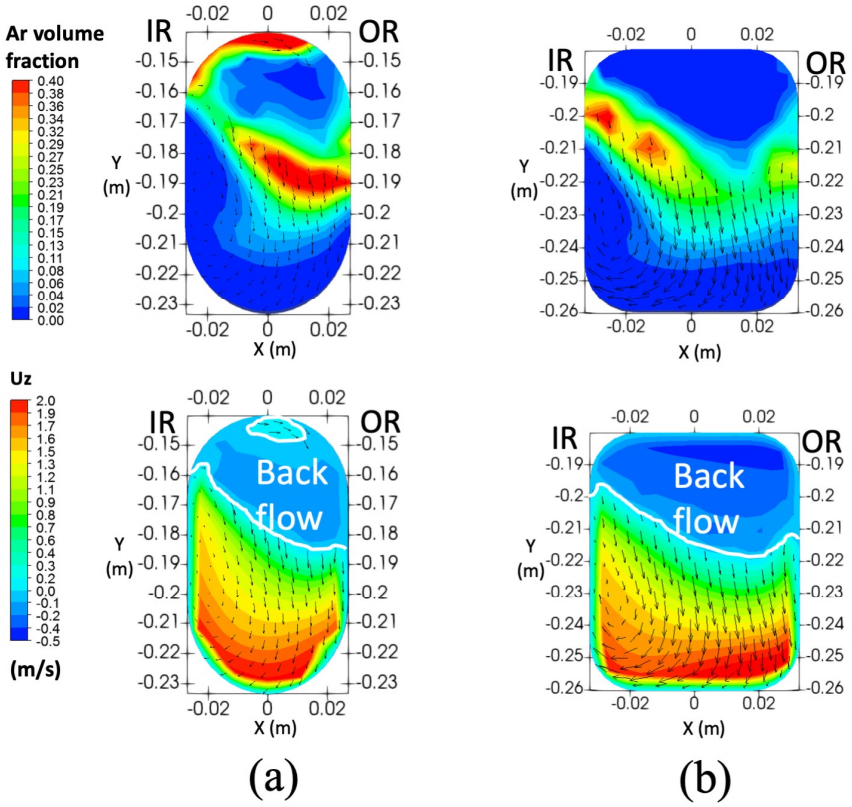


Figure 3.5 Argon volume fraction and steel flow velocity at SEN ports: (a) Case 1 and (b) Case 2

Table 3.2 Jet characteristics

Parameters	Symbols	Case 1	Case 2
Weighted average liquid velocity at the nozzle port in x-direction	$\bar{U}_{xs}$	0.15m/s	0.14m/s
Weighted average liquid velocity at the nozzle port in y-direction	$\bar{U}_{ys}$	0.58m/s	0.35m/s
Weighted average liquid velocity at the nozzle port in z-direction	$\bar{U}_{zs}$	1.56m/s	1.54m/s
Weighted average turbulence kinetic energy at the nozzle port	$\bar{K}$	$0.11\text{m}^2/\text{s}^2$	$0.22\text{m}^2/\text{s}^2$
Weighted average turbulence dissipation rate at the nozzle port	$\bar{\epsilon}$	$3.05\text{m}^2/\text{s}^3$	$6.18\text{m}^2/\text{s}^3$
Vertical jet angle	θ_{yz}	20.49°(downward)	12.73°(downward)
Horizontal jet angle	θ_{xz}	5.44° to IR	5.27° to IR
Jet speed	U_{jet}	1.67m/s	1.58m/s
Back-flow zone fraction	η	30.91%	34.86%

3.6 Strand Flow Results

Figure 3.6 compares the flow patterns in the strand for Cases 1 and 2. At a quarter longitudinal vertical-plane, the side views in this figure show that flow recirculation is generated near where the straight part of the caster transitions into the curved region. Below this flow recirculation, both cases exhibit flow stagnation regions, where downward flow starts to develop. These stagnant regions where particles are not washed away from the solidification front, have greater chance for particles to be captured into the solidifying steel shell there. Note that Case 2 has deeper stagnation regions compared to Case 1. In addition, near the narrow face (vertical plane 1 m far from the mold center), downward flow is stronger with Case 2. Thus, it is expected that more particles go down deep into the strand for Case 2, resulting in more particle capture defects, deeper into the strand. The coinciding location of this stagnant region with the curved region of caster is significant, because large particles are more easily captured into the inside radius strand due to their upward buoyancy force.

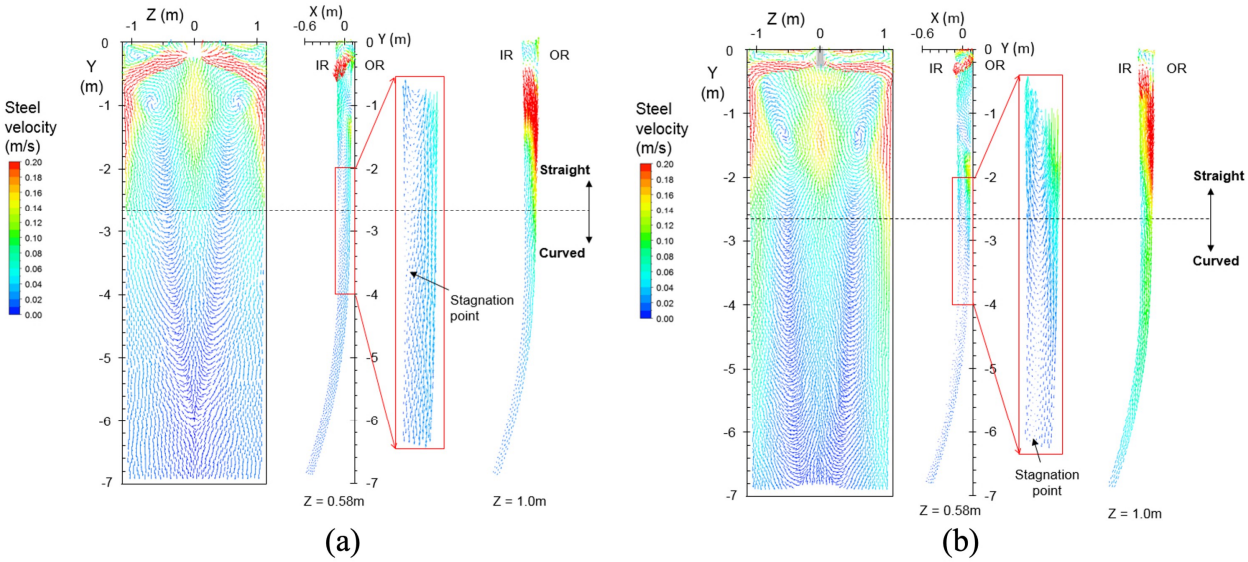


Figure 3.6 Flow patterns in the strand for (a) Case 1 and (b) Case 2

Figure 3.7 shows closeups of the flow patterns in the upper part of the strand. Case 2 exhibits a nearly classic double-roll flow pattern. The higher argon gas fraction and higher (more upward) port angle for Case 2 causes the jet to impinge higher up on the narrow face than for Case 1. Case 1 appears to have a more complex flow pattern, where the upward flow of gas leaving the nozzle ports causes strong upward flow along the SEN walls (Figure 3.7 (a) label 3). Upon reaching the surface, this flow bends towards the narrow faces, meeting the flow from the double-roll flow towards the SEN. Previous work suggests that such complex flow often leads to more surface defects [72, 73].

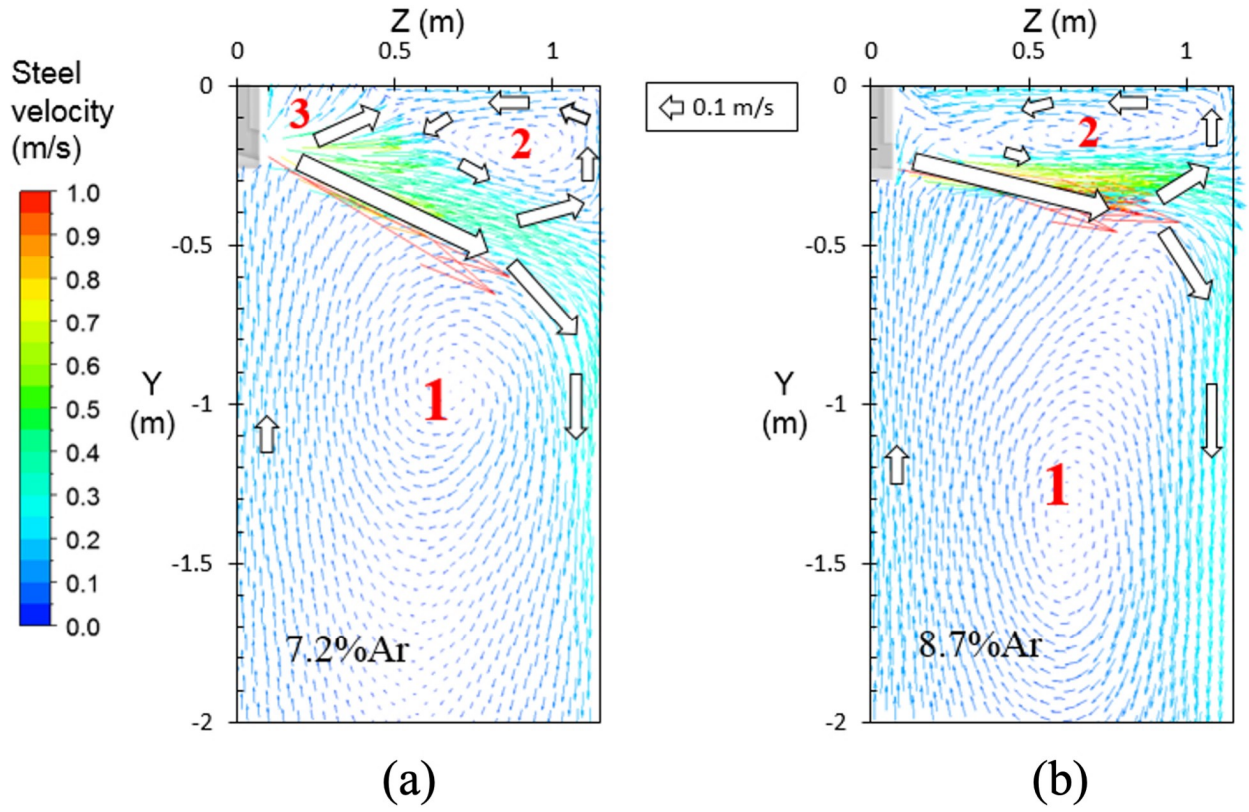


Figure 3.7 Jet flow in the upper strand for (a) Case 1 and (b) Case 2

3.7 Flow Model Validation with Nail-Board Measurements

Figure 3.8 and Figure 3.9 (see page 26) compare the predicted surface flow within a horizontal plane 10 mm below the slag/molten steel interface with nail board measurements of surface flow velocity and direction in the commercial steel plant, for case 1 and case 2. The predicted flow patterns roughly match with the measured patterns. The trends of flow generally towards the SEN for both cases, maximum velocity midway between SEN and NF for both cases, and cross flow slightly more towards the inside radius in Case 2, all match with the measurements.

However, the modeled and measured surface flow velocity magnitudes differ for both cases, especially the cross-flow velocity near the SEN. This might be due to the uniform bubble size and steady flow assumed in the Eulerian-Eulerian model. In addition, the instantaneous surface flow pattern measurement from just one nail board plant test in this highly-variable transient-flow region of the caster could differ from the long time average behavior, even during steady continuous casting [74, 75].

Efforts are made in Chapter 4 to make the surface flow match measurements qualitatively.

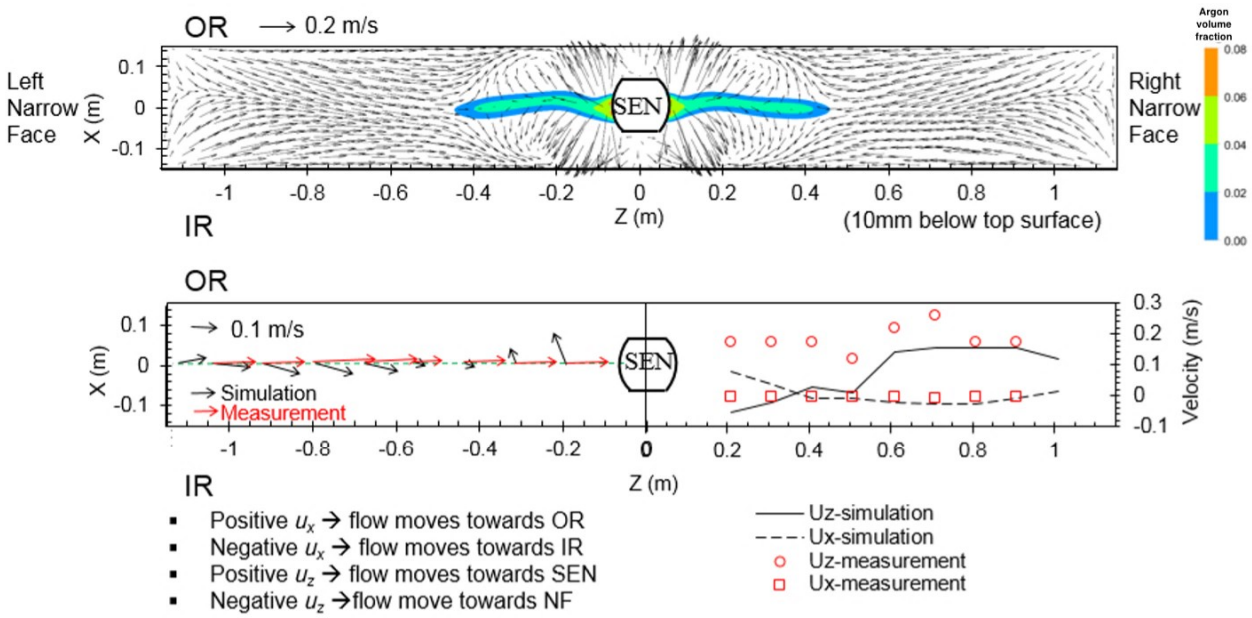


Figure 3.8 Top surface velocities comparing CFD predictions and nail board measurements for case 1

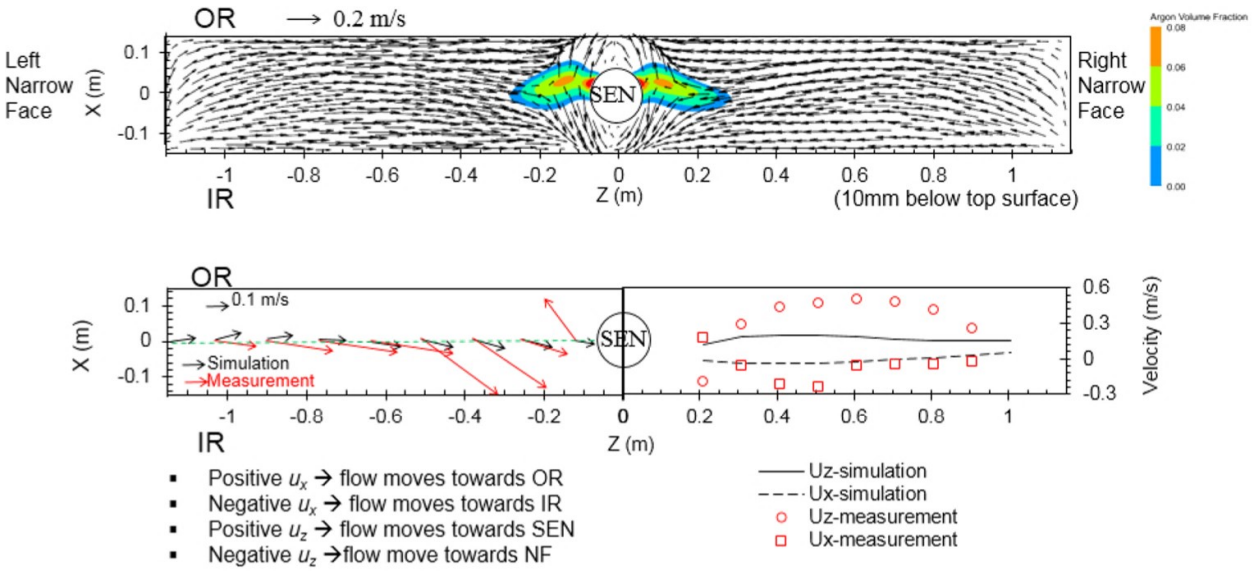


Figure 3.9 Top surface velocities comparing CFD predictions and nail board measurements for case 2

3.8 Conclusions

A modeling approach has been developed to investigate multiphase turbulent flow in the nozzle and strand during steady continuous casting of steel slabs. The predictions have been validated via plant measurements of nail board dipping tests of surface velocity. Results for two cases are compared, where

nozzle geometry and casting conditions produce 1) good quality and 2) bad quality where internal defects were much more prevalent in the plant.

The upwards flow from SEN 1 indicates that large bubbles can be more easily transported into the top slag layer, thus giving better steel quality.

To better understand the complex phenomena investigated in this work, and to lessen the capture of large particles, much further work is needed, using both improved computational models and measurements, which is described in Chapter 4.

CHAPTER 4

MODELING OF MULTIPHASE FLOW, SUPERHEAT DISSIPATION, PARTICLE TRANSPORT AND CAPTURE IN A VERTICAL AND BENDING CONTINUOUS CASTER

Modified from a paper published by the MDPI journal, Processes [8].

Mingyi Liang[9,10], Seong-Mook Cho[11], Xiaoming Ruan [12], Brian G. Thomas [13 14]

4.1 Abstract

A new model of particle entrapment during continuous casting of steel is presented, which includes the effects of multiphase flow from argon gas injection and thermal buoyancy from superheat in the strand. The model simulates three different capture mechanisms, including capture by solidified hooks at the meniscus, entrapment between dendrites, and engulfment by the surrounding of large particles. The fluid flow and bubble capture results are validated with plant measurements, including nail board dipping tests and ultrasonic tests, respectively, and good agreement is seen. Results suggest that the superheat has a negligible effect on the flow in the mold region. However, higher (30 K) superheat causes a more complex flow in the lower strand by creating multiple recirculation zones due to the thermal buoyancy effects. This causes less penetration deep into the strand, which leads to fewer and shallower particle captures. Lower (10 K) superheat may enable significant top surface freezing, leading to very large internal defect clusters. Lower superheat also leads to deeper meniscus hooks, which sometimes (0.003%) capture large (1 mm) bubbles. Capture bands occur near the transition line from vertical to curved, due to the downward fluid velocity balancing the particle terminal velocity, enabling capture in the relative stagnation region beneath the longitudinal recirculation zone. These findings agree with plant observations.

4.2 Introduction

In the continuous casting of steel, fluid flow in the nozzle and strand can capture detrimental particles, such as inclusions from upstream ladle refining and tundish, mold slag entrainment, and argon bubbles, which are often injected into the nozzle in order to lessen nozzle clogging [63]. Although most of the particles escape through the top-surface slag layer, some can be captured beneath the meniscus hooks near the top surface [14, 15]. Everywhere at the solidification front, small particles may flow in between primary

[8]Reproduced with the permission from the co-authors: Seong-Mook Cho and Xiaoming Ruan
[9]Graduate student at the Colorado School of Mines
[10]Primary researcher and author
[11]Professor, Department of Metallurgical Engineering, Pukyong National University
[12]Steelmaking Research Department, Research Institute Baoshan Iron & Steel Co., Ltd.
[13]Author for correspondence
[14]Professor, Mechanical Engineering Department, Colorado School of Mines

dendrite arms and become entrapped by the solidifying steel. Large particles may become surrounded by growing dendrites to become "engulfed" in the solidifying shell. All of these captured particles can lead to permanent surface and internal defects, such as pinholes, slivers, blisters, and expensive rejects, or unsightly and dangerous quality problems in the final product [16]. Thus, there is a great incentive to understand the complex phenomena of particle transport and capture by multiphase fluid flow and solidification, to optimize the casting conditions, modify the nozzle geometries, and improve the steel quality.

Many researchers have investigated the transport and capture of argon bubbles and non-metallic inclusions by assuming instant capture when a bubble or inclusion touches the solidification front, referred to as the "simple criterion" [11, 17–21, 76, 77]. Although this method is valid for particles smaller than the primary dendrite arm spacing (PDAS), it overpredicts the capture of particles too large to fit between the primary dendrite arms, which are often washed away back into the flow.

Efforts have been made to better predict the capture of bubbles larger than PDAS. Liu et al. [21] assume that bubbles with diameter larger than 500 μm are not captured by the solidification front, but this method naturally underpredicts the small but important capture fraction of such large bubbles. Liu and Li [78] assume that inclusions are captured by the solidification front where the liquid fraction is less than 0.6 [22, 78], and in addition to that, Chen et al. [22] requires that the fluid speed is less than 0.07 m/s. Thomas and coworkers [18, 19, 23, 24, 33, 46] developed an advanced capture criterion which includes both the entrapment of small particles and the engulfment mechanism of large particles, based on a force balance at the solidification front. This approach has been adopted [79, 80] or modified [26] by others.

The entrapment / engulfment criterion matches well with plant step-milling measurements [24, 79, 80] and is better than the simple criterion [24]. Two other studies have applied this entrapment / engulfment criterion for both argon bubbles and inclusions on both straight and bending sections of two plant casters [6, 25]. A band of captured particles was predicted at around 1/4 and 3/4 across the slab thickness from the inner radius (IR) of the caster [6, 22, 25, 26], which agrees with Ultrasonic-Test (UT) maps of particle numbers and locations in as-cast slabs.

However, previous investigations with this entrapment / engulfment capture criterion underpredict the bubbles captured near the slab surface, possibly due to the model neglecting the subsurface hook capture [24, 27]. Hooks form during initial solidification at the meniscus and can degrade the steel quality, by entrapping argon bubbles and non-metallic inclusions beneath them while they move down, resulting in slab surface defects and slivers in the rolled product [14, 81, 82]. Several previous works investigated the effect of operating parameters on subsurface hooks [28–31]. Lee et al. developed an empirical equation to estimate the hook depth for ultra-low carbon steel [32], which confirms the expectation that increasing superheat delivery to the meniscus region or increasing casting speed has a strong effect on decreasing the

hook depth [32]. Recently, a hook capture mechanism was implemented into a computational fluid dynamics (CFD) model, by considering a bubble as captured if it stays long enough in a hook zone [33]. Therefore, both this hook capture mechanism and the entrapment / engulfment capture criterion are implemented in the current study to predict argon bubble capture in the continuous casting process.

As another important phenomenon, superheat can potentially affect the particle capture by changing the molten-steel flow pattern in the strand region due to thermal and solutal buoyancy. Only a few previous studies have investigated superheat transport in computational flow models of continuous casting of steel [12, 13, 22, 78]. These models show that most of the superheat is removed in the mold region, or just below [12, 13]. The temperature of the molten steel in the mold decreases continuously with distance along the path travelled by the flowing jet, with the lowest temperatures found at the meniscus region, near the narrow faces (NF) and near the submerged entry nozzle (SEN), for a typical double-roll flow pattern [12, 13]. Huang et al. [13] shows that superheat has negligible effect on the mold flow. Shi et al. [83] measured the temperature in a mold, and showed that the Reynold-Averaged Navier–Stokes (RANS) model can match the measurements. However, no previous model has modeled fluid flow deep into the caster, where velocity is slower, and the importance of superheat is unknown. Superheat is known to affect the internal quality, owing to its strong effect on microstructure. Lower superheat produces less centerline segregation and related quality problems [84–86]. However, its effect on particle capture has received little investigation.

To study the two-phase fluid flow of molten steel and argon gas in the nozzle and mold with different casting conditions and its subsequent impact on particle capture, many studies have been conducted using three-dimensional CFD models [6, 13, 22, 24, 26, 27, 37, 39–41, 77, 78, 78–80, 87–90], including Eulerian-Eulerian (E-E) [6, 39, 40, 90] Eulerian-Lagrangian [6, 22, 24, 26, 27, 37, 77–80], and Large Eddy Simulation (LES) [22, 27, 78, 87–89] approaches. It is important to include two-way coupling in multiphase flow simulations because the gas and molten steel flow affect each other's behavior. For instance, increasing the argon flow rate can change the mold flow from a classic double-roll to a single-roll flow pattern with the surface flow reversing to flow from the SEN towards the NF [39, 40, 91]. Excessive gas fractions have been observed to lead to oscillating and asymmetric flow [43, 44]. Jin et al. developed a RANS k-ϵ model coupled with a Discrete Phase Model (DPM) and the entrapment / engulfment capture criterion, to investigate the argon bubble capture in a continuous casting nozzle and a straight strand. The turbulent dispersion of each bubble can be successfully mimicked by the Random Walk Model (RWM), which has been used in many previous studies [19–21, 23, 37, 45, 47, 48, 78, 90, 92]. Results show that 85 pct of small (< 0.08 mm) bubbles are captured and only a small fraction of large bubbles is captured (< 0.02 pct) [45].

Building on previous efforts [6, 24], a new, computationally-efficient (more efficient than RANS-DPM or LES-DPM two-way coupled model) modeling methodology has been developed, to quantify the effect of superheat on the multiphase molten steel flow pattern and argon bubble capture in continuous slab casting. This new model system features an improved prediction of bubble size distributions and efficient post-processing to decrease the required number of injected bubbles.

4.3 Computational Models and Solution Procedure

The multiphase flow of molten steel with argon gas, superheat transport, and temperature distribution in the molten steel, particle transport, and particle capture via three different mechanisms was simulated using a system of computational models.Figure 4.1 shows a flow chart of the computational model system.

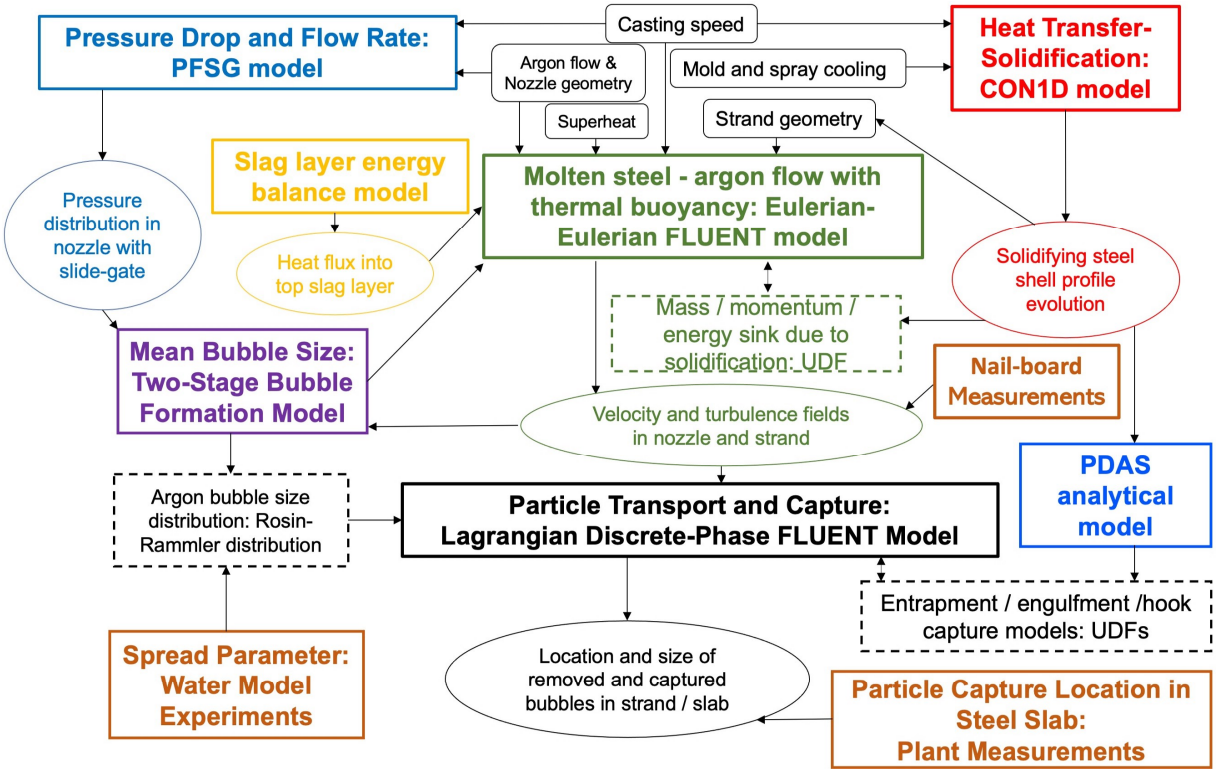


Figure 4.1 Model flow chart

First, the mean bubble size needed by the multiphase flow model was predicted using a model of pressure distribution and flow rate within a slide-gate flow delivery system (PFSG) model [64] to find the local pressure, and a two-stage bubble size model for gas injection into downward flowing metal [65]. With the predicted mean bubble size and a spread parameter from the curve fitting of a water model experiment, the bubble size distribution was obtained. A heat transfer and solidification model (CON1D) was used to

estimate the shell thickness profile down the strand, which was characterized with a solidification constant, and used to build the strand liquid pool geometry, implement the mass/momentum/energy sinks due to the steel solidification, and predict the PDAS size evolution.

Then, the multiphase flow in the nozzle and liquid pool inside the strand was computed with an E–E model (Ansys Fluent), including the effects of thermal buoyancy with a fully coupled heat transfer model of superheat transport. A simple energy balance model of the top slag layer was developed to calculate the heat flux lost from the top surface of the molten steel. Next, different bubble sizes were injected into the strand flow field and tracked using a Lagrangian DPM model (Ansys Fluent). This model includes hook/entrapment/engulfment capture mechanisms (fluent user-defined functions (UDFs)) to determine the capture of argon bubbles. A model of the PDAS size (down the caster) was needed to evaluate the capture mechanism. Finally, a new post-processing method was applied to calculate the particle capture fractions and compare them with measurements in the slab samples.

Details of all these computational models are provided in the following subsections.

4.3.1 Initial Bubble Size Models: PFSG and Two-Stage Model

The mean bubble size used for the Eulerian-Eulerian model is predicted by the Pressure distribution and Flow rate within Slide Gate flow delivery system (PFSG) model [64] and a two-stage bubble size model [65], as described in Chapter 3. With the predicted mean bubble size, a Rosin-Rammler distribution is assumed for argon bubbles [93]. The volume fraction of argon contained in the bubbles which have diameter less than d_{pi} is defined by F(d_{pi}):

$$F(d_{pi}) = \frac{Q_h(d_p < d_{pi})}{Q_h} = 1 - e^{-(\frac{d_{pi}}{d_{p,mean}})^n} \tag{4.1}$$

where the spread parameter, n, is taken as 3.5 based on curve fitting of the bubble size measurements in a water model [94].

The bubble injection rate and volume fraction of different bubble sizes are shown in Figure 4.2:

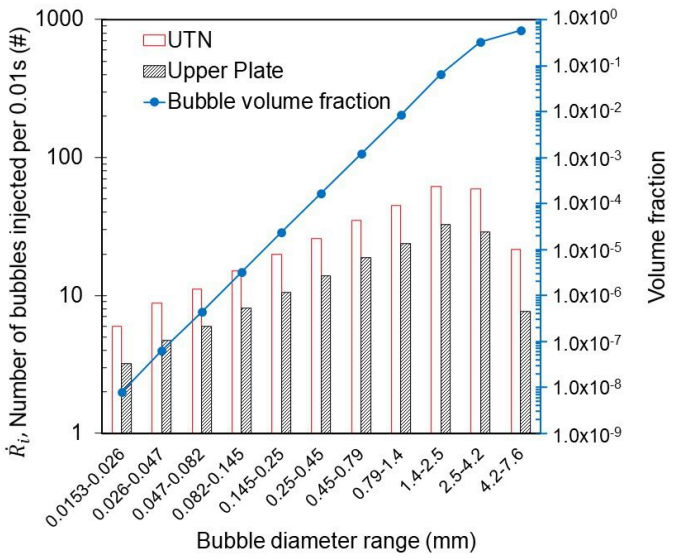


Figure 4.2 Injection rate and volume fraction of bubbles

4.3.2 Heat Transfer and Solidification: CON1D

A heat transfer CON1D model and a solidification empirical constant are used to estimate the shell thickness profile using equation 3.2, which is used to build the strand liquid-pool geometry, implement the mass / momentum / energy sinks due to the steel solidification, and predict the PDAS size revolution. The solidification constant K is taken as 3.2 mm/$\sqrt{s}$. Figure 4.3 shows that the empirical shell thickness compares reasonably with the plant data. A reasonable agreement is seen.

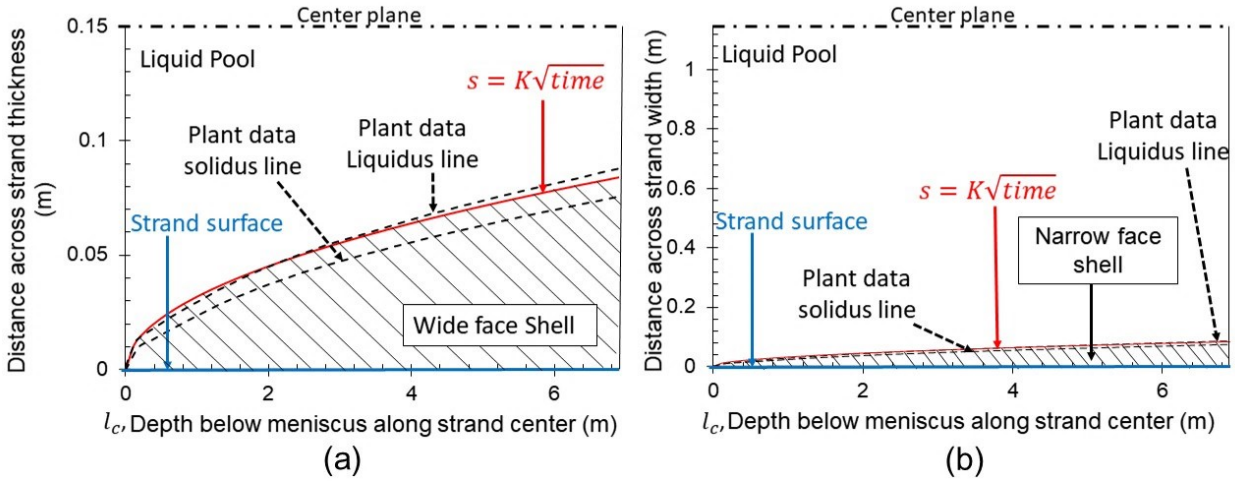


Figure 4.3 Shell thickness profile: (a) On wide face (WF) (b) On narrow face (NF)

4.3.3 Two-Phase Flow: Eulerian-Eulerian Model

A three-dimensional steady-state E-E RANS model is employed to simulate the multiphase flow of molten steel and argon gas in both the nozzle and the liquid pool of the caster, using the commercial finite-volume software ANSYS FLUENT [67]. UDFs are used to account for mass, momentum and energy sinks around the liquid pool boundary due to the steel solidification [18, 68]. The governing equations of the E-E model solve separate continuity and Navier-Stokes equations for each phase. The continuity equation of each phase is as follows:

$$\frac{\partial}{\partial x_i}\alpha_q \rho_q u_{iq} = S_{q-mass-sink} \tag{4.2}$$

where the subscript q is either s or g, denoting the steel or argon gas phase, and the sum of the volume fraction of each phase is 1 in each cell (α_s+α_g=1); u_{iq} is the steel or argon gas velocity tensors; ρ_q is the density of steel or argon; $S_{q-mass-sink}$ is the mass sink term for liquid or gas phase, which is applied to the cells adjacent to the solidification front to account for the mass loss due to the liquid phase transitioning into the solid steel. The mass sink term for steel can be calculated by the equation below:

$$S_{s-mass-sink} = \rho_s v_c A_c V_{cell}^{-1} \tag{4.3}$$

where A_c is the projected area of wall-adjacent cell face in the casting direction (m^2), and V_{cell} is the wall-adjacent cell volume (m^3). A_c can be calculated by $A_c = A_f \times sin\theta_c$, where A_f is the wall-adjacent face area (m^2), and θ_c is the angle between the face and the casting direction. Previous work calculates θ_c along the straight part of the strand [17–19, 24, 27, 33, 68]. Based on Equation 3.2, the following equation is derived for θ_c on the curved strand:

$$\theta_c = arcsin(\frac{\partial s}{\partial l_c}) = arcsin(0.5K\sqrt{\frac{v_c}{l_c}}) \tag{4.4}$$

The argon-gas mass sink term is only applied to the top surface to simulate the process of argon gas escaping into the top slag layer, because the bubble diameter in the E-E model is set as the calculated mean bubble diameter $d_{p,mean}$, which is large. So, the continuous-phase bubbles tend to float up into the slag layer immediately after they flow through the nozzle port. The mass sink term for the argon gas can be calculated by the equation below:

$$S_{g-mass-sink} = \rho_g v_{gy} A_y \alpha_g V_{cell}^{-1} \tag{4.5}$$

where v_{gy} is the gas-phase vertical velocity towards the top surface, A_y is the top-wall-adjacent cell face area (m^2), and α_g is the gas volume fraction in the top-wall-adjacent cell.

The Navier-Stokes equations for momentum balance of each phase are:

$$\alpha_s \frac{\partial \rho_s u_{is} u_j}{\partial x_j} = -\alpha_s \frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}[(\mu_s + \mu_{ts})(\frac{\partial u_{is}}{\partial x_j} + \frac{\partial u_{js}}{\partial x_i})] + \alpha_s F_{Bi,g} - R_{igs} + \sum F_{is} + F_{Bi,t} + S_{is-momentum-sink} \tag{4.6}$$

$$\alpha_g \frac{\partial \rho_g u_{ig} u_j}{\partial x_j} = -\alpha_g \frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}[(\mu_g + \mu_{tg})(\frac{\partial u_{ig}}{\partial x_j} + \frac{\partial u_{jg}}{\partial x_i})] + \alpha_g F_{Bi,g} - R_{igs} + \sum F_{ig} + S_{ig-momentum-sink} \tag{4.7}$$

where P is the pressure shared by all phases, and μ_s and μ_g are the dynamic viscosity of the molten steel and argon gas. The turbulent viscosity μ_{ts} and μ_{tg} are calculated by the standard k-ϵ model. The gas buoyancy force, $F_{Bi,g}$, caused by the density difference between the liquid and gas phase, is $(\rho_s - \rho_g)g_i$, where g_i is the gravitational acceleration (-9.81 m/s^2 in y direction). The thermal buoyancy force, $F_{Bi,t}$, is $(\rho_s(T) - \rho_0)g_i$, where $\rho_s(T)$ is the temperature-dependent density of the steel at a given location relative to the reference density, as discussed in the next section. The momentum sink terms for molten steel and argon gas, $S_{is-momentum-sink}$ and $S_{ig-momentum-sink}$, can be calculated based on their mass sink terms: $S_{iq-momentum-sink} = S_{q-mass-sink} u_{iq}$.

The additional forces on the liquid and gas phases, $\sum F_{is}$ and $\sum F_{ig}$, are defined as follows:

$$F_{is} = -F_{ig} = F_{iVM} + F_{iD} = \overbrace{0.5\alpha_g \rho_s (\frac{Du_{is}}{Dt} - \frac{Du_{ig}}{Dt})}^{\text{virtual mass force, } F_{iVM}} \tag{4.8}$$

$$\underbrace{- \frac{3}{4}(\frac{C_{D1}}{d_p})\rho_s \alpha_g \mid u_{ig} - u_{is} \mid (u_{ig} - u_{is})}_{\text{drag force}, F_{iD}} \tag{4.9}$$

where F_{iVM} is a virtual mass force, F_{iD} is a drag force, C_{D1} is a drag coefficient, and d_p is the argon bubble diameter.

A source term for the interphase force, R_{igs}, is defined as:

$$R_{igs} = \alpha_g(1 - \alpha_g)\rho_g [\frac{\overbrace{C_{D1} Re_p}^{f_{drag}}}{24}](u_{ig} - u_{is}) \overbrace{[\frac{\rho_g d_p^2}{18\mu_s}]^{-1}}^{\tau_p} \tag{4.10}$$

where f_{drag} is a drag function, τ_p is the particle relaxation time which represents the time scale for a particle to respond to the changes in the surrounding flow, and Re_p is the particle Reynolds number due to the relative difference between the particle and local fluid velocities, which is defined by $\frac{\rho_s |u_{ig} - u_{is}| d_p}{\mu_s}$. The Tomiyama drag model [69] is chosen. This model can be applied to a wide range of bubble-shape regimes

by including the Eotvos number Eo in the drag coefficient (C_{D1}) calculation:

$$C_{D1} = max(min(\frac{24}{Re_p}(1 + 0.15Re_p^{0.687}), \frac{72}{Re_p}), \frac{8}{3}\frac{Eo}{Eo+4}) \tag{4.11}$$

where $Eo = \frac{g(\rho_s - \rho_g)d_{p,eq}^2}{\sigma}$ which represents the ratio of buoyancy to surface tension force, σ is the surface tension (1.157 N/m) between argon gas and molten steel, and $d_{p,eq}$ is the sphere volume-equivalent bubble diameter.

For the turbulence modeling, the standard k-ϵ model is adopted [95]. The two-equation model attempts to predict the turbulence by solving equations for two variables, turbulent kinetic energy (k) and its dissipation rate (ϵ). The model is given by

$$\frac{\partial \rho_m k}{\partial t} + \frac{\partial \rho_m u_{im} k}{\partial x_i} = \frac{\partial}{\partial x_j}[(\mu_m + \frac{\mu_{tm}}{\sigma_k})\frac{\partial k}{\partial x_j}] + G_{k,m} - \rho_m \epsilon \tag{4.12}$$

$$\frac{\partial \rho_m \epsilon}{\partial t} + \frac{\partial \rho_m u_{im} \epsilon}{\partial x_i} = \frac{\partial}{\partial x_j}[(\mu_m + \frac{\mu_{tm}}{\sigma_\epsilon})\frac{\partial \epsilon}{\partial x_j}] + \frac{\epsilon}{k}(C_{1\epsilon}G_{k,m} - C_{2\epsilon}\rho_m \epsilon) \tag{4.13}$$

where $G_{k,m}$ is the production of turbulent kinetic energy. The turbulent Prandtl numbers for k and ϵ, σ_k and σ_ϵ, are 1.0 and 1.3 respectively. Other constants have following values: $C_{1\epsilon}$=1.44, $C_{2\epsilon}$=1.92, C_μ=0.09.

The mixture density, ρ_m, mixture turbulent viscosity, μ_m, and mixture velocity, u_{im}, are computed from $\rho_m = \alpha_g \rho_g + \alpha_s \rho_s$, $\mu_m = \alpha_g \mu_g + \alpha_s \mu_s$, and $u_{im} = \frac{\alpha_g \rho_g u_{ig} + \alpha_s \rho_s u_{is}}{\alpha_g \rho_g + \alpha_s \rho_s}$. The mixture turbulent viscosity μ_{tm} can then be calculated by $\mu_{tm} = \rho_m C_\mu \frac{k^2}{\epsilon}$. The turbulent viscosity μ_{ts} and μ_{tg} in Equations (4.6) and (4.7) can be calculated from $\mu_{tg} = \frac{\rho_g}{\rho_m}\mu_{tm}$ and $\mu_{ts} = \frac{\rho_s}{\rho_m}\mu_{tm}$.

4.3.4 Heat Transfer: CFD Thermal Model

The following steady, scalar, 3D energy equation is solved for the steel phase, in the liquid pool that comprises the strand domain:

$$\frac{\partial \alpha_s \rho_s u_{is} h_s}{\partial x_i} = \frac{\partial}{\partial x_i}(k_s + k_{ts})\frac{\partial T_s}{\partial x_i} + S_{hs} \tag{4.14}$$

where h_s is the sensible energy of the steel and can be calculated by $\int_{T_{ref}}^{T_{local,s}} c_{ps} dT_s = c_{ps}(T_{local,s} - T_{ref})$, where T_{ref} is the reference temperature, 298.15K, and $T_{local,s}$ is the predicted local temperature (K). The thermal capacity c_{ps} and conductivity k_s are in Table 4.1. The turbulent thermal conductivity, k_{ts}, can be calculated by $\frac{c_{ps}\mu_{ts}}{P_{rt}}$, where P_{rt} is the turbulent Prandtl number, 0.85. The energy sink term, S_{hs}, is $S_{s-mass-sink}c_{ps}(T_{local,s} - T_{ref})$, and accounts for advection heat transfer across the solidification front.

The steel density is calculated [67] from the temperature results, based on its volumetric expansion coefficient β:

$$\rho_s(T) = \rho_0 \times [1 - \beta \times (T - T_0)] = 7000kg/m^3 \times [1 - 0.000100K^{-1} \times (T - 1826K)] = 8278.2 - 0.7T \tag{4.15}$$

where ρ_0 is the density at the reference temperature, T_0. This density is needed to evaluate thermal buoyancy term, $F_{Bi,t}$, in Equation 4.6. Solutal buoyancy would also affect the flow near to the solidification front, where rejected solute from segregation may accumulate. In this work, solutal buoyancy was neglected, owing to the small segregation associated with the low carbon and low alloy content of the steels considered here.

4.3.5 Energy Balance Model for Heat Flux into Top Slag Layer

Slag is constantly added onto the molten steel to cover the top surface, serving as an adiabatic layer to prevent oxidation and freeze due to heat loss into the atmosphere. Therefore, the heat flux from molten steel to top slag layer is crucial in obtaining an accurate superheat dissipation result. A simple energy balance model is used for this purpose. Figure 4.4 (see page 38) shows a schematic of a control volume of the top slag layer (inside the dashed box). Solid slag powder is added to the top of the mold and gradually melt into sintered state and liquid phase, forming a three-layer stratified flow. Under steady state, energy coming in by adding solid slag ($Q_{in,top}$) and conduction from bottom hotter steel ($Q_{in,bottom}$) has to be equal to that getting out of this control volume, by convection with far-field atmosphere ($Q_{out,top}$) and liquid slag dropping down from the gap between copper mold and steel shell ($Q_{out,bottom}$).

The energy balance equation can be written as:

$$Q_{in,top} + Q_{in,bottom} = Q_{out,top} + Q_{out,bottom} \tag{4.16}$$

$$Q_{in,top} = c_{p,pow}(T_{top} - T_\infty)\rho_{slag}v_y A \tag{4.17}$$

$$Q_{out,top} = h_s A(T_{top} - T_\infty) \tag{4.18}$$

$$Q_{out,bottom} = [\overbrace{c_{p,pow}(T_{slag,sol2} - T_{top})\rho_{slag}v_y}^{\text{powder layer heat flux}} + \overbrace{c_{p,sin}(T_{slag,liq} - T_{slag,sol2})\rho_{slag}v_y}^{\text{sintered layer heat flux}} + \overbrace{c_{p,liq}(T_{steel,liq} - T_{slag,liq})\rho_{slag}v_y}^{\text{liquid layer heat flux}}] \tag{4.19}$$

where A is the strand cross-sectional area and can be calculated by $t_s \times w - s$, where t_s and w_s is the thickness and width of the strand (0.3 m and 2.3 m in this thesis). $Q_{in,bottom}/A$, the heat flux from hot

molten steel into the liquid slag layer, can be solved by solving Equations (43) - (46). $c_{p,pow}$, $c_{p,sin}$, $c_{p,liq}$ are the heat capacity of the powder, sintered slag, and liquid slag, 1000 $J/kg \cdot K$, 2750 $J/kg \cdot K$, and 3000 $J/kg \cdot K$, respectively [96]. T_∞, T_{top}, $T_{slag,sol2}$, $T_{slag,liq}$ are ambient temperature, powder top surface temperature, slag solidus temperature, and slag liquidus temperature; they have values as 300 K, 585 K, 1300 K, and 1400 K respectively [96]. h_s is the heat transfer coefficient between powder surface and far-field atmosphere, 2.7 $W/m^2 \cdot K$. Note that the surface temperature is taken as that of trial 112 from A. Akhtar [96], and the top surface heat transfer coefficient is taken as an averaged value of trial 501 and 517 from A. Akhtar [96], because these trials have similar carbon content as the steel in this study. Slag density, ρ_{slag}, is set as 2600 kg/m^3.

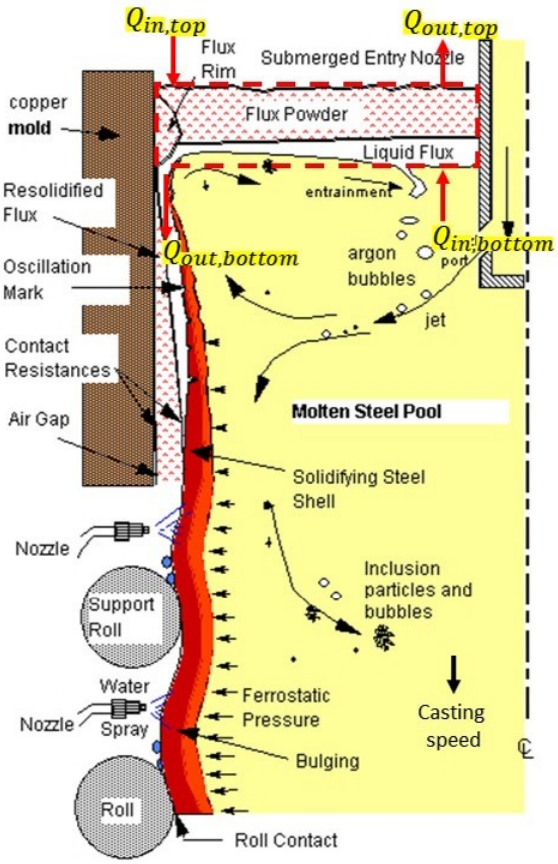


Figure 4.4 Slag layer energy balance [5]

Another unknown is v_y (m/s), the downward velocity of slag consumption, i.e. the downward velocity of powder as it melts into liquid slag. v_y can be calculated by:

$$v_y = \frac{\overbrace{(c_{OM} + c_{lub}) \times f \times 2(t+w)}^{C_t}}{1000 \rho_{slag} t w} \tag{4.20}$$

where f is the estimated oscillation frequency (cycles/s) and can be computed by $93 \times v_c = 93 \times 0.01$ m/s =0.93 $cycles/s$ [96]. C_t is the slag consumption rate in g/s. c_{OM} and c_{lub} are slag consumption rate per unit length per unit cycle due to oscillation mark and lubrication ($g/m \cdot cycle$), and they can be calculated by an empirical relation developed by Shin et al. [97]:

$$c_{OM} = 2.5 \times 10^{-2} \times \rho_{slag} \times k_e^{1.43} \times t_n^{0.389} \times v_c^{-1.49} (\sqrt{\frac{2\sigma_{ss}}{\triangle \rho g}})^{0.556} \tag{4.21}$$

$$c_{lub} = 0.507 \times e^{3.59 \times t_p} \tag{4.22}$$

where k_e is an empirical constant, 17.8. σ_{ss} is the surface tension between slag and steel, 1.3 N/m. $\triangle \rho$ is the density difference between steel and slag, 4527 kg/m^3. t_n and t_p are the negative and positive strip time. They can be calculated by the following equations:

$$t_n = \frac{1}{\pi f} cos^{-1}(\frac{v_c}{\pi s_0 f}) \tag{4.23}$$

$$t_p = T - t_n \tag{4.24}$$

where s_0 is the oscillation stroke, 7 mm [96]. T is oscillation cycle time (s). The Equations 4.16) - 4.19 can be solved all together first to get v_y. With v_y, then Equations 4.20 - 4.24 can be solved to get the heat flux from molten steel into slag top layer. The results are: $t_n = 0.36s$, $t_p = 0.71s$, $c_{OM} = 5.8 g/m \cdot cycle$, $c_{lub} = 6.54 g/m \cdot cycle$. $C_t = 59.68 g/s$, and $v_y = 3.3 \times 10^{-5} m/s$. $\frac{Q_{in,top}}{A} = 24.70 kW/m^2$, $\frac{Q_{out,top}}{A} = 0.77 kW/m^2$, $\frac{Q_{out,bottom}}{A} = 188.60 kW/m^2$, and finally, the heat flux from molten steel into top slag layer is $\frac{Q_{in,bottom}}{A} = \frac{Q_{out,top}}{A} + \frac{Q_{out,bottom}}{A} - \frac{Q_{in,top}}{A} = 164.67 kW/m^2$.

4.3.6 Lagrangian DPM Model

The Lagrangian DPM Model is used to simulate particle argon bubbles transport in the molten steel pool in the caster in a one-way-coupled way with the steel-argon two-phase Eulerian-Eulerian flow in SEN and strand.

The following momentum balance equation is solved for each bubble with velocity u_{ip}:

$$\frac{Du_{ip}}{Dt} = \underbrace{\frac{18\mu}{\rho_p d_p^2}\frac{C_{D2}}{24}(u_{is}-u_{ip})\frac{\overbrace{\rho_s d_p \mid u_{ip}-u_{is} \mid}^{Re_p}}{\mu}}_{F_{iD}} + \underbrace{0.5\frac{\rho_s}{\rho_p}(\frac{Du_{is}}{Dt}-\frac{Du_{ip}}{Dt})}_{F_{iVM}} + \underbrace{\frac{\rho_s}{\rho_p}u_{ip}\frac{\partial u_{is}}{\partial x}}_{F_{iP}} + \underbrace{\frac{g(\rho_p-\rho_s)}{\rho_p}}_{F_{iB}} \tag{4.25}$$

where F_{iVM} is virtual mass force, per unit bubble mass an additional force required to accelerate the surrounding fluid when the bubble is accelerated. F_{iP} and F_{iB} are pressure gradient force and buoyancy/gravity force per unit bubble mass. Re_p is the particle Reynolds number. F_{iD} is the drag force per unit bubble mass, and the drag coefficient, C_{D2}, in DPM model was computed based on the drag law proposed by Moris *et.al*[98].

$$C_{D2} = a_1 + \frac{a_2}{Re_p} + \frac{a_3}{Re_p^2} \tag{4.26}$$

$$a_1, a_2, a_3 = \begin{cases} 0, 24, 0 & \\ 3.690, 22.73, 0.0903 & 0.1 < Re_p < 1 \\ 1.222, 29.1667, -3.8889 & 1 < Re_p < 10 \\ 0.6167, 46.50, -116.67 & 10 < Re_p < 100 \\ 0.3644, 98.33, -2778 & 100 < Re_p < 1000 \\ 0.357, 148.62, -47500 & 1000 < Re_p < 5000 \\ 0.46, -490.546, 578700 & 5000 < Re_p < 10000 \\ 0.5191, -1662.5, 5416700 & Re_p > 10000 \end{cases} \tag{4.27}$$

When one-way coupled with Eulerian-Eulerian RANS model, a RWM is used to account for the effect of small eddies on the bubbles. The isotropic RWM is used to approximate the chaotic dispersion of particles due to turbulence. In this model, a Gaussian distributed random velocity fluctuation, u_i' is generated using Equation 4.28, which assumes isotropic turbulence everywhere in the fluid domain, in proportion to the local turbulence level. The random velocity fluctuation is applied on each argon bubble at each time step as if it is kicked towards a random direction by a small force. In this way, the effect of small eddies on bubble movement is considered into the RANS model.

$$u_i' = \xi\sqrt{2k/3} \tag{4.28}$$

where ξ is a standard normally-distributed random number (i.e. mean = 0 and standard deviation = 1). The random number is not changed until the interaction time of the eddy and a particle t_{inter} firstly reached the eddy life-time t_e or the eddy cross time t_{cross} (the time required for a particle to pass the eddy). These two time scales are defined as:

$$t_e = -0.15\frac{k}{\epsilon}\ln\gamma \tag{4.29}$$

$$t_{cross} = -\tau_p \ln\left[1 - \frac{L_e}{\tau_p \mid u_{is} - u_{ip} \mid}\right] \tag{4.30}$$

where γ is a uniformly-distributed random number from 0 to 1. This randomness of γ generates a random eddy life time that increases in regions with high local turbulence and little dissipation. L_e is the eddy length scale.

4.3.7 Particle Capture Model: FLUENT UDFs

Equations describing the hook, entrapment, and engulfment capture mechanisms are coded into UDFs and were used in this study to determine the capture of argon bubbles. The flow chart of this capture process is shown in in Figure 4.5.

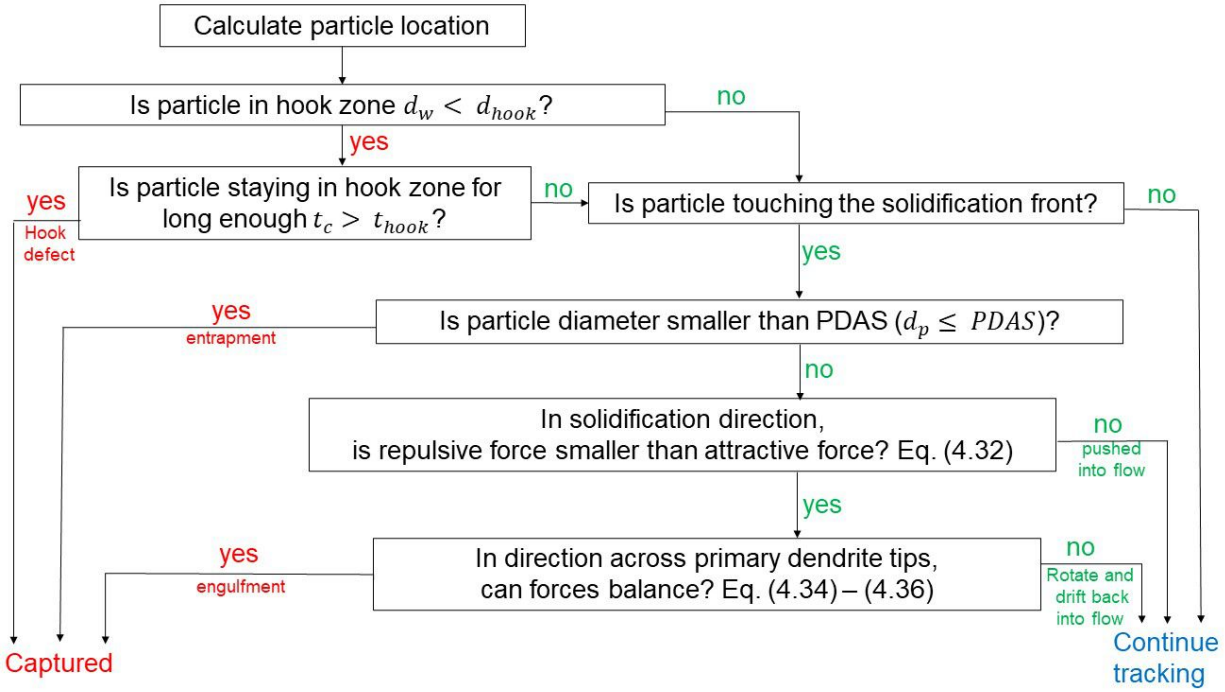


Figure 4.5 Capture Flow Chart

Particles can be captured as they rise beneath the solidified meniscus hooks, which are modeled by the hook capture mechanism [33]. The hook zone is defined as a region below the hook and above the solidifying shell, as shown in Figure 4.6. There are three possible fates of a particle entering the hook zone: (1) touch the moving-down hook and become captured, (2) touch the solidification front and evaluate

the criteria for the entrapment/engulfment mechanisms, and (3) flow out of the hook zone and back into the bulk flow.

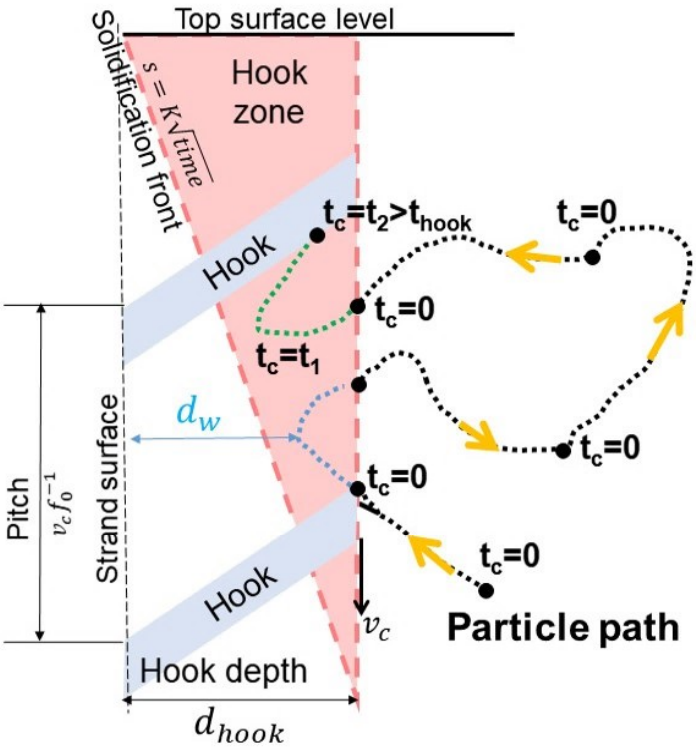


Figure 4.6 Hook capture mechanism schematic

The hook depth d_{hook} is defined as the horizontal distance between the hook tip and the slab surface. Lee et al. conducted experiments on ultra-low carbon steel samples to quantify the relation between the casting parameters and the hook depth, and an equation was proposed to predict the hook depth as a function of mold oscillation frequency f_0, mean level fluctuation L_F, and solidification temperature of slag powder $T_{slag,sol1}$ [32]. In this study, this equation was applied for low-carbon steel with $f_0 = 68$ cycles/min, $L_F = 1.8$ mm, and $T_{slag,sol1} = 1150$ °C. The results are hook depths of 5.6 mm with low (10 K) superheat and 4.5 mm with high (30 K) superheat. The real hook depth was rarely this deep, possibly because the empirical equation was developed from experiments on ultra-low carbon steel, which is more susceptible to hook formation. However, the calculated deep hooks can be used to investigate worst-case scenarios regarding hook capture and compare with the entrapment/engulfment mechanisms to study the importance of the hook capture mechanism.

To prevent the solidifying shell from sticking to the mold, the mold oscillates at a frequency. Each oscillation cycle allows a new meniscus hook to form, and the shortest distance between every two adjacent hooks (pitch), Δh_{hook}, can be calculated by: $\Delta h_{hook} = v_c f_0^{-1}$. Then $\Delta h_{hook} = 0.0100\text{m/s} \times \frac{68\ cycles}{60\ s} = 11.3$ mm.

When the distance between a particle and the strand surface d_w is less than d_{hook}, the particle enters the hook zone. The hook capture mechanism assumes that if a particle stays long enough in the hook zone, it will be captured. The critical time for a particle to be captured, t_{hook}, is approximated as the time required for a particle to travel half of a pitch:

$$t_{hook} = \frac{\Delta h_{hook}}{2v_c} = 0.5 f_0^{-1} \tag{4.31}$$

In this study, $t_{hook} = 0.44$ s. A cumulative timer t_c is set up to track the time that each particle entering the hook zone stays there, as shown in Figure 4.6. If t_c ¿ t_{hook}, the particle will be considered as captured by the hook. When a particle is not in or escapes the hook zone, t_c is reset to 0 s.

The entrapment / engulfment particle capture criterion [6, 18, 19, 24, 25, 27, 33, 46] is applied to predict the capture of particles touching the solidification front. Bubbles smaller than the PDAS can flow between arms and become entrapped, as shown in Figure 4.7 (a) (see page 44). Larger particles may become engulfed into the solidification front, if the particle remains stationary at the solidification front, i.e. the forces cannot rotate the bubble around the dendrite tip, as shown in Figure 4.7 (b) (see page 44). The latter occurs according to a balance of 6 different forces at the solidification front: Drag force (F_D), buoyancy/gravity force (F_B), life force(F_L), lubrication force,(F_{Lub}) Van der waals force (F_I), and surface tension gradient force (F_{Grad}).

If the forces on a bubble touching solidification front do not satisfy Equation 4.32, meaning that the net force is pointing away from the solidification direction, the bubble is pushed back into the flow and be continuously tracked.

$$F_L - F_{D,\chi} - 2(F_{Lub} - F_{Grad} - F_I)cos\theta < 0 \tag{4.32}$$

where χ denotes the solidification direction, and θ is defined as:

$$\theta = arcsin(\frac{PDAS}{2(R_p + r_d)}) \tag{4.33}$$

where R_p is the particle radius and r_d is the dendrite tip radius.

Otherwise, if Equation 4.32 is satisfied, meaning that the net force is towards the solidification front and bubble is pushed against it, the next step is to check if the force in η direction (tangential to

solidification direction) can rotate the bubble around the primary dendrite tips by following equations:

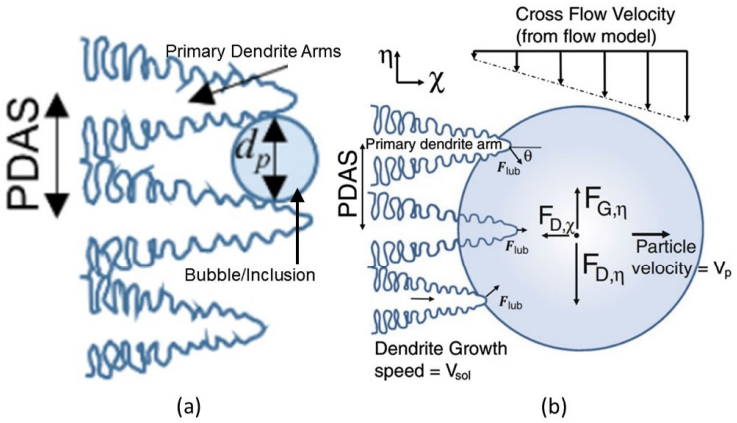


Figure 4.7 Capture criterion schematic: (a) Entrapment in between primary dendrites (b) Engulfment by growing solidification shell

if $F_{D,\eta}$ and $F_{B,\eta}$ are in the same direction:

$$(F_{D,\eta} + F_{B,\eta})cos\theta + (F_L + F_{D,\chi})sin\theta < (F_{Lub} - F_{Grad} - F_I)sin2\theta \tag{4.34}$$

if $F_{D,\eta}$ and $F_{B,\eta}$ are in the opposite direction, and $F_{D,\eta} \geq F_{B,\eta}$:

$$(F_{D,\eta} - F_{B,\eta})cos\theta + (F_L + F_{D,\chi})sin\theta < (F_{Lub} - F_{Grad} - F_I)sin2\theta \tag{4.35}$$

if $F_{D,\eta}$ and $F_{B,\eta}$ are in the opposite direction, and $F_{D,\eta} < F_{B,\eta}$:

$$(F_{B,\eta} - F_{D,\eta})cos\theta + (F_L + F_{D,\chi})sin\theta < (F_{Lub} - F_{Grad} - F_I)sin2\theta \tag{4.36}$$

If one of Equations 4.34 - 4.36 is satisfied, the bubble will not be able to rotate around the dendrite tips. Instead, the bubble stay rest on the solidification front and be engulfed. Otherwise, the bubble rotates and gets drifted back into the flow.

The buoyancy / gravity force (F_B), virtual mass force (F_{VM}), and pressure gradient force (F_P) have same definition as in the DPM model. The drag force which points tangentially against the movement of a bubble is defined as:

$$F_D = \frac{1}{8}\pi\rho d^2 C_{D3}|u_s - u_p|(u_p - u_s) \tag{4.37}$$

where C_{D3} is the drag coefficient, defined as $C_{D3} = f_{Re_p}(\frac{24}{Re_p})$. Re_p is particle Reynolds number, and f_{Re_p} is the correction factor due to a finite particle Reynolds number [99]: $f_{Re_p} = (1 + 0.15{Re_p}^{0.687})$.

The lift force, caused by the shear flow, is calculated by [100]:

$$F_L = \frac{-9}{\pi}\mu_s R_p^2(u_s - u_p)sgn(G)\left[\frac{|G|}{v_s}\right]^{\frac{1}{2}} J^u \tag{4.38}$$

where v_s is molten-steel kinematic viscosity. G is the wall normal velocity gradient. J^u is determined by a dimensionless parameter, ε, defined as $\varepsilon = sgn(G)\frac{\sqrt{|G|v}}{u_s - u_p}$. J^u can be determined by the following equation when $0.1 \leq \varepsilon \leq 20$ [101]:

$$J^u = 0.6765\{1 + tanh[2.5log_{10}\varepsilon + 0.19]\}\{0.667 + tanh[6(\varepsilon - 0.32)]\} \tag{4.39}$$

For other ε values, $J^u = -32\pi^2[sgn(\varepsilon)\varepsilon]^5 log\frac{1}{\varepsilon^2}$

The lubrication force is caused by molten steel moving into the gap between the bubble and the dendrite tip. This force can be written as [102, 103]:

$$F_{lub} = 6\pi\mu V_{sol}\frac{R_p^2}{h_o}(\frac{r_d}{r_d + R_p})^2 \tag{4.40}$$

where V_{sol} is the solidification velocity, and h_o is the distance between the dendrite tip and the particle.

The Van der waals force is calculated by [104]:

$$F_I = 2\pi\Delta\sigma_o\frac{r_d R_p}{r_d + R_p}\frac{a_o^2}{h_o^2} \tag{4.41}$$

where $\Delta\sigma_o = \sigma_{sp} - \sigma_{sl} - \sigma_{pl}$, where σ_{sp}, σ_{sl}, and σ_{pl} are surface tensions for solid shell/bubble, solid shell/ liquid steel, and particle/liquid steel, respectively. a_o is the atomic diameter of molten steel.

The surface energy gradient force is caused by the rejection of solute near the solidification front, mainly sulfur. Carbon and sulfur are the two main solutes rejected near the solidification front. Carbon is more important for solutal buoyancy force as it changes the steel density, while sulfur is more important for surface tension force. The near-solidification side of bubble has higher sulfur content than the side away from solidification front, which makes the surface tension higher on the hot side than the other cold half. The net surface tension force pushes the bubble towards the solidification front and can be written as [18, 105]:

$$F_{Grad} = -\frac{m\beta_G\pi R_p}{\xi_1^2}\left\{\frac{(\xi_1^2 - R_p^2)}{\beta_G^2}ln\left[\frac{(\xi_1 + R_p)[\alpha_G(\xi_1 + R_p) + \beta_G]}{(\xi_1 - R_p)[\alpha_G(\xi_1 + R_p) + \beta_G]}\right] + \frac{2R_p}{\alpha_G}\right\} + \frac{m\beta_G^2\pi R_p}{(\xi_1\alpha_G)^2}ln\left[\frac{\alpha_G(\xi_1 + R_p) + \beta_G}{\alpha_G(\xi_1 - R_p) + \beta_G}\right] \tag{4.42}$$

where $\alpha_G = 1 + n_1 C_o$, $\beta_G = n_1 r_d (C^* - C_o)$, $\xi_1 = R_p + r_d + h_o$. m and n_1 are empirical constants, 0.17 J/m^{2}and 844 (mass%)$^{-1}$, respectively. C^* can be calculated by $\frac{V_{sol} r_d}{D_s} = \frac{C^* - C_o}{C^*(1-k_1)}$, where C_o is the sulfur content of steel, D_s is the diffusion coefficient of sulfur in steel, and k_1 is the distribution coefficient. Details of this criterion and forces are provided elsewhere [18, 19].

Bubbles exiting the domain outlet is considered as captured as well, because the domain is 7-m-long, and it is unlikely for a bubble traveling that deep to float up and be removed by the top slag layer. The particle capture UDFs used are in Appendix A.

To validate this model system, the model results are compared with plant measurements of the nail-board test and UT maps.

4.3.8 Primary Dendrite Arm Spacing

The particle capture models require the PDAS profile as a function of the distance below meniscus. The PDAS is important because particles smaller than the PDAS are easily entrapped in between the steel dendrites when they contact the solidification front. Several empirical correlations exist in the literature for dendrite arm spacings, based on measurements for different steel grades and cooling rates [66, 106–109].

The dendrite arm spacing increases with lower carbon content, as shown by various researchers [107, 108, 110]. In the solidifying steel strand, the PDAS also increases greatly with distance beneath the slab surface (also down the strand for a given casting speed). The shell growth can be characterized by the solidification constant, $K(m/\sqrt{s})$. The cooling rate, solidification front velocity, $V_{sol}(m/s)$, and temperature gradient at the solidification front, $G_T(^\circ C/m)$, can be extracted from K, the steel liquidus temperature $T_{steel,liq}(^\circ C)$, steel thermal diffusivity $\alpha_{th}(m^2/s)$, and casting time $t_{cas}(s)$ according to classic analytical solutions for solidification [111]. The figure showing the current model correlation for the PDAS was presented in previous literature [6], which is based on the relation from Y. Quan [18] and Kurz and Fisher [112]:

$$PDAS = 4.3\lambda V_{sol}^{-\frac{1}{4}} G_T^{-\frac{1}{2}} \tag{4.43}$$

where λ is a constant by curve fitting the experimental data, 7.2×10^{-13} m^3 $^\circ C^2/s$ and 4.3×10^{-13} m^3 $^\circ C^2/s$ for the WF and NF respectively, based on measurements [6]. V_{sol} and G_T can be calculated by [111]: $V_{sol} = \beta_{th}\sqrt{\frac{\alpha_{th}}{t_{cas}}}$ and $G_T = \frac{T_{steel,liq} + \triangle T_{sup} - T_{int}}{e^{\beta_{th}^2} erf(\beta_{th})\sqrt{\pi \alpha_{th} t_{cas}}}$, where T_{int} is the interfacial temperature between solid and liquid steel (1000 $^\circ C$), $\alpha_{th} = \frac{k_s}{\rho_s c_{ps}}$, and $\beta_{th} = \frac{K}{2\sqrt{\alpha_{th}}}$. The PDAS used in this study on the wide face is the blue line as shown in Figure 4.8.

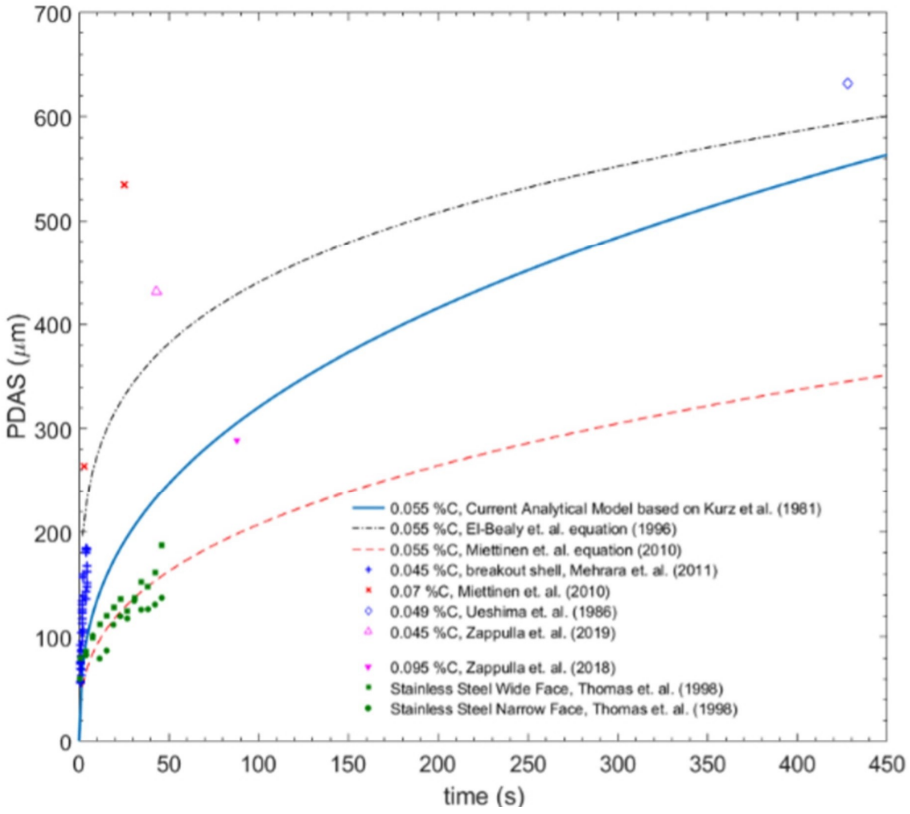


Figure 4.8 PDAS revolution profile vs. casting time on the wide faces

4.4 Flow Model Domain and Mesh

The flow model domain, including the complete slide-gate opening, nozzle, and 7-m-long liquid pool in the strand is shown in Figure 4.9. Due to its symmetry about its center plane, only half of the flow domain is modeled. The caster has a 2.665 m vertical upper section, followed by bending. To construct the liquid pool domain, multiple cross-sectional planes (perpendicular to the casting direction) were generated at different distances below the meniscus along the strand, with more planes created near the meniscus. The liquid pool width-by-thickness sketches were built on each plane based on the shell thickness profile in Figure 4.3, then the liquid pool domain was created by connecting all of the two-dimensional sketches using the ANSYS Workbench Design Modeler. In this way, the need to model the solidifying steel shell was avoided, and the domain shape was constant throughout the simulation.

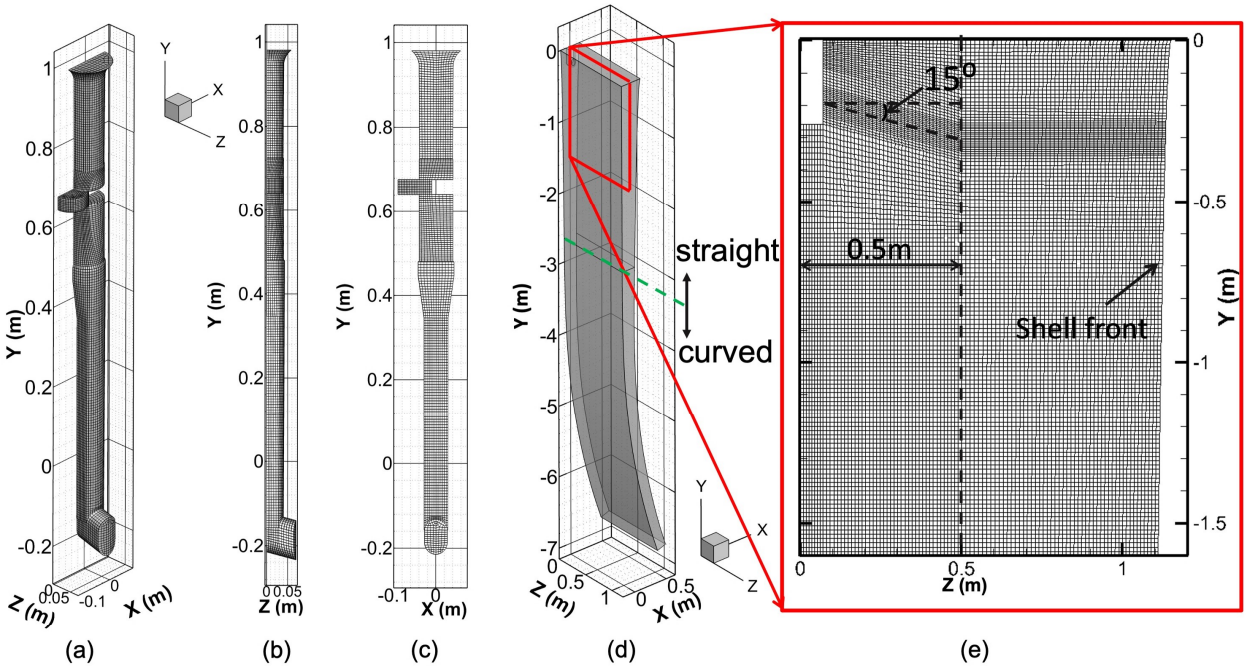


Figure 4.9 Flow domain and mesh (a) Nozzle 3D view, (b) front view, (c) nozzle center plane, side view, (d) Liquid pool, and (e) Liquid pool center plane

A mesh of structured hexahedral cells was created in the ANSYS Workbench meshing and included ∼20,000 cells for the nozzle domain and ∼1.1 million cells for the liquid pool in the strand. Numerical diffusion increases when the mesh has an angle relative to the flow direction and is worst for 45° [113], so the mesh was constructed to align cell faces parallel to the nozzle jet flow direction to minimize numerical diffusion. Convergence for the steady-state simulation on a wide caster (width $>$ 2 m) is difficult because the nozzle jet must travel further to reach the NF, which destabilizes the flow for this wide (2.3 m) caster, so first-order upwind was used for discretization of the advection terms in Equations (4.6), (4.7) and (4.12)–(4.14). The standard wall function was used for flow in the wall-boundary cells [95]. This allowed a coarser mesh near the boundaries, due to its reasonable treatment of the turbulent boundary layer. The maximum boundary cell size in the strand of the current mesh has a Y^+ value of less than the recommended maximum of 300 [67].

Figure 4.9 (a)–(c) shows the nozzle domain mesh. Figure 4.9 (d), (e) shows the mold domain geometry and the upper 1.6-m-high mesh on the center plane. The mesh near the mold inlet is refined due to the finer mesh on the nozzle outlet and the 1-on-1 data interpolation of velocity and turbulence from the nozzle outlet to the mold inlet. Within 0.5 m from the symmetry plane, the mesh was constructed with a 15° downward angle to help align the mesh with the flow direction, and thereby reduce false diffusion. Beyond 0.5 m towards the NF, the jet tends to lose its downward momentum and starts to flow

horizontally. The mesh was then made parallel to the z direction until the steel shell front.

4.5 Boundary Conditions

The boundary conditions for nozzle flow (step 1), iso-thermal strand flow (step 2), strand flow with superheat (step 3), and bubble transport and capture (step 4) simulations are given in Figure 4.10.

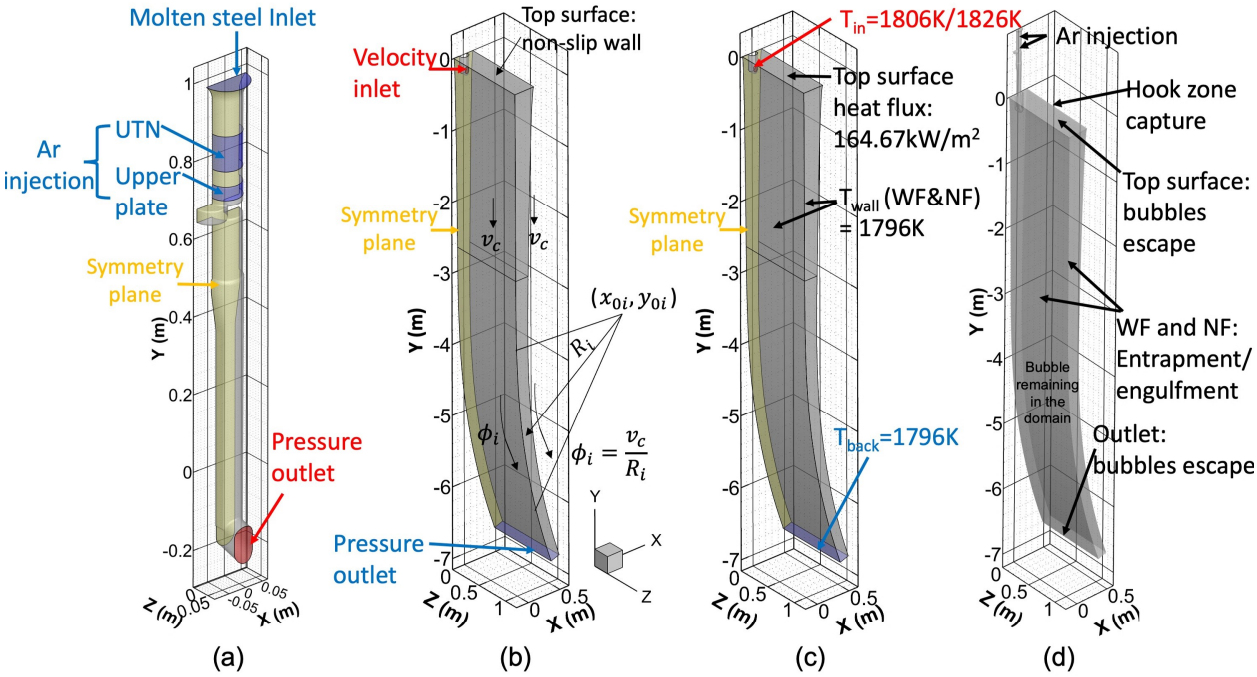


Figure 4.10 Boundary conditions for (a) Step 1: nozzle flow, (b) Step 2: strand flow, (c) Step3: heat transfer, and (d) Step 4: bubble transport and capture

Step 1 is to obtain the isothermal flow in the nozzle. Because of the short residence time of molten steel in the nozzle ($\frac{nozzle\ length}{avg.velocity} = \frac{1.2\ m}{1.4\ m/s} = 0.86$ s), the heat transfer in the nozzle is negligible. Then, the nozzle outlet velocity, turbulent kinetic energy, and turbulent dissipation rate are interpolated onto the inlet plane (nozzle port) (Figure 4.10 (b)), and in step 2, the isothermal flow is calculated in the strand. In step 3, the temperature field is first estimated based on the steady-state isothermal flow pattern from step 2. Then the steel density is updated based on this initial temperature field and a fully-coupled temperature-multiphase fluid-flow calculation is performed to find the final E–E flow field. Finally, in step 4, discrete argon bubbles are injected into the E–E flow field from step 3, and bubble capture is determined with the capture models described in Section 4.3.7.

In step 1, a constant mass flow rate of 24.15 kg/s is applied at the inlet; the turbulent kinetic energy k and its dissipation rate ϵ are fixed at 10^{-5} m^2/s^2 and 10^{-5} m^2/s^3. The argon gas volume flow rate and mean bubble size are used for the argon gas injection through both the UTN and upper plate, at the

locations shown in Figure 4.10 (a). The argon gas is injected at 0.015 m/s through the UTN and 0.016 m/s through the upper plate, based on the hot argon volume flow rate divided by the inlet area. The nozzle wall has 0.00100 m roughness and has a non-slip boundary condition.

In steps 2 and 3, the top surface is approximated as non-slip for the molten steel flow, due to the large viscosity of the slag layer. Other surfaces (the WF and NF) are also set as non-slip walls with the mass and momentum sink UDFs acting in their adjacent cells. The thermal model in step 3 has an energy sink UDF. The narrow and wide faces are moving in the casting direction at the casting speed. As shown in Figure 4.10 (b), to simulate this wall velocity in the ANSYS FLUENT, the straight part of the narrow and wide faces are given a vertical velocity 0.0100 m/s (casting speed) in the casting direction. The curved part is given a rotational velocity, ϕ_i (rad/s), about the arc center (x_{0i}, y_{0i}), based on the curved arc radius R_i (m) and the casting speed v_c (m/s), where i = 1, 2, ..., 7, as there are 7 arcs with different radii: $\phi_i = \frac{v_c}{R_i}$. A pressure outlet condition is chosen over the nozzle outlet and the strand domain bottom, according to the ferrostatic pressure head of the molten steel ($\rho_s g h$), where h is the vertical distance below the meniscus. The turbulent kinetic energy and its dissipation rate are set as $k = 10^{-5}$ m^2/s^2 and $\epsilon = 10^{-5}$ m^2/s^3 for the reversed flow (in case the flow re-enters the domain) throughout steps 1–3. In step 3 (Figure 4.10 (c)), the inlet temperatures are 1806 K (10 K superheat) and 1826 K (30 K superheat). The top-surface heat flux into the liquid slag layer is 164.7 kW/m^2. The narrow face and wide faces, representing the solidification front, are set at the steel liquidus temperature, 1796 K, as is any flow re-entering the domain. Throughout steps 1–3, the gradient of every variable on the center symmetry plane is 0.

Finally, in step 4, the domains and flow patterns in the nozzle (step 1) and the strand (step 3) are combined, and the trajectories of discrete bubbles are then computed by the DPM model in a one-way-coupled manner. A certain number of bubbles are injected for each of the 11 bubble sizes in Figure 4.2. As its trajectory is integrated, each injected bubble can either escape into the top surface (slag layer) or be captured by the hook mechanism, entrapment, or engulfment in the hook zone, the vertical part of the strand, or later, deep in the curved strand. Particles that either remain in the domain after 500 s of simulation or exit the domain outlet are assumed to be captured deep in the strand, as shown in Figure 4.10 (d). The time step size is 0.005 s to satisfy the Courant number criterion.

4.6 Numerical Details

All the governing equations are discretized and solved by a Gauss-Seidel linear equation solver and an Algebraic Multigrid (AMG) method in ANSYS FLUENT with its Coupled algorithm [67], on a staggered grid with zero velocities everywhere as initial conditions. This algorithm simultaneously solves the equations of velocity corrections of each phase and pressure correction [114] and is very efficient in

steady-state simulations [67].

Simulation of the nozzle flow (Step 1) is conducted on a Dell computer tower with the Intel(R) Xeon(R) E5-2609 v3 @1.90GHz CPU, 64.0GB RAM, and 10 computational nodes. The simulation results can be obtained by ∼30 minutes with all residuals reaching 10^{-4}. For the strand flow (Step 2 and 3), the high-performance computer Mio is used with 48 computational nodes, and the results are considered converged when all residuals reached 10^{-7}. the computational time for the mold flow is ∼ 5 hours. The same computational power is used for step 4, and the simulation time is 500 s which takes ∼1 day. After 500 s, more than 99.98% of the bubbles end up either escaping into the top slag layer and outlet, or being captured by hooks, entrapment, and engulfment.

4.7 Post-Processing of Bubble Capture

When running the model to predict particle capture, transient simulations should be conducted for sufficient time to obtain statistically significant capture at every location and particle size. However, particles can be captured at a given location of the caster for a long time, because they can penetrate faster than the casting speed, and can persist in the strand for many residence times. The average residence time in the 7-m-long computational domain can be estimated by:

$$t_r = \frac{m_s}{\dot{m}} \tag{4.44}$$

where m_s is the total steel mass inside the computational domain, and $\dot{m}$ is the mass flow rate of the molten steel. In this study, $t_r = \frac{m_s}{\dot{m}} = \frac{9907kg}{24.15kg/s} = 410s$. Thus, conventional particle simulations require significant computational time.

Moreover, some bubble sizes are rare. For example, in this work, Figure 4.2 shows that 0.02 mm bubbles are injected at only 900 #/s, compared with 5400 #/s 0.6 mm bubbles (2.5 million particles per residence time). Considering that there are other 9 sizes, injecting particles at every time step until reaching statistically-significant capture results is a very inefficient method.

To overcome this problem, a new methodology has been developed to find the number of bubbles with diameter d_{pi} captured, N_{ci}, during the sample casting time, based on the corresponding number of bubbles with diameter d_{pi} captured in the simulation N_{si}:

$$N_{ci} = \dot{R}_i \frac{l_s N_{si}}{v_c I_i} \tag{4.45}$$

where $\dot{R}_i$ is the injection rate (#/s) of bubbles with diameter d_{pi} into a full-domain nozzle, as shown in Figure 4.2, l_s is the measured slab sample length (m), and I_i is the total number of bubbles (with

diameter d_{pi}). These bubbles are injected as a burst, at the beginning of the DPM simulation. Then, N_{ci} can be compared with the plant measurements, such as those in the UT samples of this work. This method allows any I_i number of particles of a given size to be simulated and compared with measurements, so long as this number is large enough to be statistically significant. Previous work shows that at least 2,500 particles are necessary to obtain accuracy within ±3% [90], and increasing the number of particles improve the accuracy. The accuracy is defined by comparing the particle capture fraction with several runs and calculate their percentage difference. So, in this work, I_i = 30,000 particles are chosen for each of the 11 sizes, to achieve an accuracy better than ±3%.

The overall capture fraction $C_{f,i}$ for bubbles with diameter d_{pi} is defined as:

$$C_{f,i} = \frac{N_{h,i} + N_{ent,i} + N_{eng,i} + N_{o,i} + N_{r,i}}{I_i} \tag{4.46}$$

where $N_{h,i}$, $N_{ent,i}$, and $N_{eng,i}$ are the number of bubbles captured by the hook, entrapment and engulfment capture mechanisms. $N_{o,i}$ and $N_{r,i}$ are the number of bubbles exiting the domain outlet and still remaining in the domain at the end of simulation.

The capture rate per meter of slab $\dot{R}_{sl,i}$ and per second $\dot{R}_{s,i}$ for bubbles with diameter d_{pi} can be calculated by:

$$\dot{R}_{sl,i} = \dot{R}_{s,i}\frac{1}{v_c} = \dot{R}_i \times C_{f,i}\frac{1}{v_c} \tag{4.47}$$

4.8 Casting Conditions

Two cases have been simulated on the same caster, for conditions in Table 4.1. The steel grade is plain low-carbon steel, typically containing 0.055% C. Argon gas is injected through the UTN and slide-gate upper plate. Only half of the domain is modeled, so the flow rates given in Table 4.1 are half of the real casting conditions. The two cases use 10 K and 30 K superheat, which is their only difference.

Table 4.1 Caster dimensions and process conditions

Parameters	Symbols	Values
Port down angle	θ	15°
Slab thickness (mm)	t_s	300
Slab width (mm)	w_s	2300
Tundish height (mm)	h_t	1400
Casting speed (m/min)	v_c	0.6
Steel flow rate (metric ton/min)	$\dot{m}$	1.45
Slide gate opening	E/E_{max}	62%
Submergence depth (mm)	d_s	140
Argon gas flow rate		
via UTN: Hot (LPM) / Cold (SLPM)	$Q_{h_utn}[Q_{s_utn}]$	11.65[2.6]
via plate: Hot (LPM) / Cold (SLPM)	$Q_{h_plate}[Q_{s_plate}]$	4.55[1.1]
Argon volume fraction	f_{Ar}	7.2%
Molten steel density (kg/m^3)	ρ_s	ρ(T) = 8278.2-0.7T
Molten steel viscosity (kg/m·s)	μ_s	0.0063
Liquidus Temperature (K)	$T_{steel,liq}$	1796
Superheat (K)	$\triangle T_{sup}$	10 / 30
Steel heat capacity (J/kg·K)	c_{ps}	680 [13]
Steel thermal conductivity (W/m·K)	k_s	26 [13]
Steel thermal expansion coefficient (K^{-1})	β	1.0x10^{-4} [13]
Ar viscosity (kg/m·s)	μ_g	2.125×10^{-5}
Ar heat capacity (J/kg·K)	c_{pg}	520
Ar thermal conductivity (W/m·K)	k_g	0.067

4.9 Flow-Model Verification and Validation

Large Eddy Simulation (LES) is more accurate than RANS modeling, because the large eddies which contain most of the turbulent energy are directly solved by LES while they are simply modelled with RANS. Here, results from the RANS model system with a ∼1.1-million-cell mesh are compared with LES of the same caster and conditions. The LES model includes a transient term in Equations (4.2), (4.6), and (4.14) and the Smagorinsky-Lilly model for the sub-grid-scale turbulent viscosity in Equations (4.6) and (4.14), instead of the standard k-ϵ turbulence model used in the RANS model. A mesh of the full domain containing ∼8-million hexahedral cells is constructed for the LES simulation, and the flow is initialized with the single-phase steady-state RANS results. Second-order upwinding is used for advection terms in the LES model, and the bounded second-order implicit scheme is used for temporal discretization. It takes ∼6 months to obtain a 71 s two-way-coupled DPM-LES fluid flow with Rosin-Rammler-distributed argon bubbles injected every time step, on a 96-node High-Performance Computer (HPC) Mio. Details of this LES simulation can be found elsewhere in Chapter 5.

The last 38 s of the LES simulation is time averaged and compared in Figure 4.11 with the steady-state RANS results with 30 K superheat. The RANS results with 10 K superheat are almost identical with those

with 30 K superheat, so are not shown.

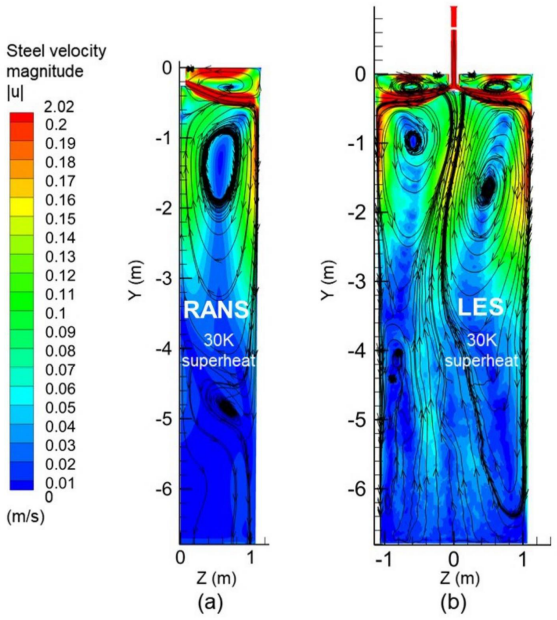


Figure 4.11 Molten steel flow pattern from (a) steady-state RANS results and (b) 38s time-averaged LES results

The average residence time for molten steel to flow through the strand top surface is estimated as the half-width (1.15 m) divided by the average velocity in the upper mold (0.15 m/s) which is ~8 s. Thus, 38 s should be enough to achieve an accurate time-averaged velocity near the top surface. Figure 4.12 compares the predicted surface velocities on a horizontal plane 10 mm below the top surface from both the RANS and the right half of LES model with nail-board measurements of the surface-flow velocity in a commercial steel plant for the same geometry and conditions.

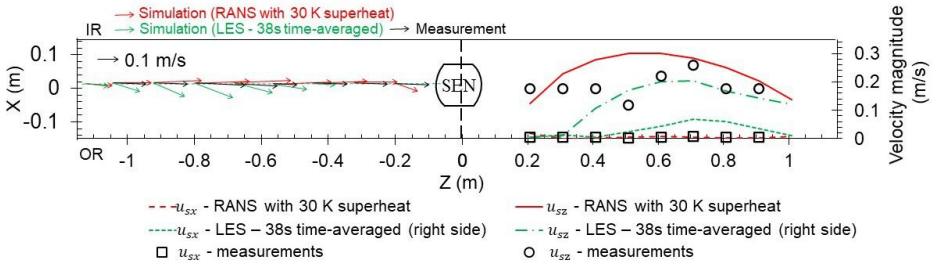


Figure 4.12 Nozzle velocity and outlet argon gas fraction

The predicted flow is a classic double-roll flow pattern, with flow towards the SEN in both models, which matches reasonably well with the plant measurements. The maximum velocity occurs near the width

/ 4 point, midway between the SEN and NF. However, the velocity towards the SEN (z-component) is somewhat overpredicted by the RANS model and underpredicted by the LES model. The LES model also predicts a strong cross flow towards the OR, which is not seen in either the RANS model or measurements. Perhaps the instantaneous surface flow pattern measurement from just one nail-board plant test in this highly-variable transient-flow region of the caster may differ from the true time-averaged behavior, even during the steady continuous casting conditions considered [74, 75].

The RANS model appears to match better with the measurements than the second-order accurate LES results, perhaps because the extra numerical diffusion from the first-order upwinding compensates for the extra turbulence of the jet entering the mold, as also observed in previous modeling of this flow system [115, 116]. Furthermore, this first-order upwinding method readily achieves a converged solution, while other RANS methods usually do not.

The LES model has other problems. The LES flow is asymmetrical in the lower strand region as shown in Figure 4.11 (b), meaning that the simulation has not reached quasi-steady state due to 71 s simulation being too short. As the average residence time in the strand, t_r, is 410 s, it would take $\sim$ 3 years with current computational power to have a fully-developed LES flow, especially deep down the caster, which is needed for accurate particle simulations. Thus, the new model system with the chosen RANS model is used in the current study, as the best compromise of efficiency and accuracy.

4.10 Nozzle Flow Results

Figure 4.13 shows the molten steel velocity results and outlet argon gas distribution in the nozzle from the new model system with the RANS model. The steel mainly exits from the lower region of the nozzle port due to its higher downward momentum, as shown by the streamlines in Figure 4.13 (a). Large recirculation zones with low flow velocity are observed just below the middle plate of the slide gate. This region has a high gas fraction, as shown in Figure 4.13 (a) and (b), which can cause bubble coalescence and gas pockets [117]. .

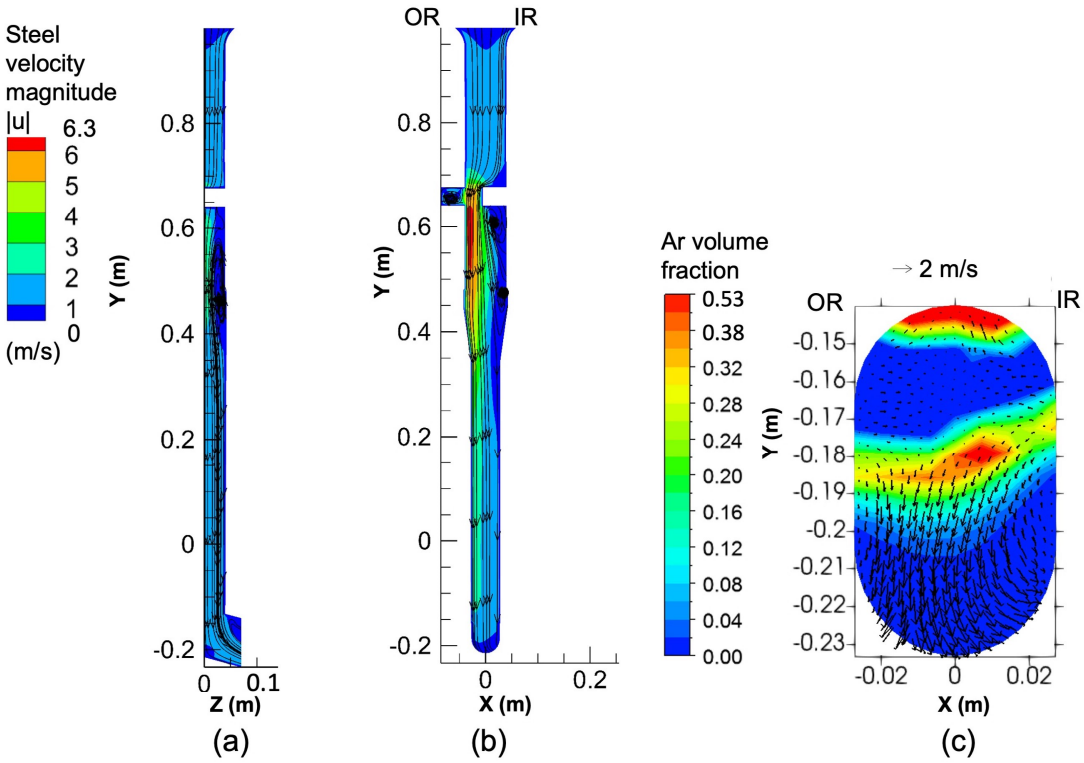


Figure 4.13 Nozzle velocity (a) front view; (b) side view; and (c) argon gas fraction in the plane of the outlet port.

Figure 4.13 (c) shows that the steel flowing through the slide gate opening causes a counterclockwise swirl through the left port, as shown by the arrows. This is caused by the movement of the slide gate towards the outer radius (OR) of the caster. The argon flows out mainly from the upper region of the nozzle port, due to its buoyancy, and from the center, due to the swirling flow momentum throwing the molten steel towards the outsides of the bottom portion of the nozzle.

The jet is defined only where there is a positive outflow from the nozzle, and details are given elsewhere [71]. The jet comes out of the port with a weighted-average speed, turbulent kinetic energy, and turbulent dissipation rate of 1.61 m/s, 0.13 m^2/s^2, and 2.17 m^2/s^3, respectively. The jet is pointing 24.06° downwards and 1.97° towards IR. The 39.43% of the upper port is back flow.

4.11 Strand Flow Results

Figure 4.14 (see page 58) and Figure 4.15 (see page 58) compares the strand flow pattern with 10 K and 30 K superheat. The upper strand flow pattern, especially near to the jet, is nearly the same for both cases. However, for flow in the lower strand region, increasing the superheat complicates the flow by

creating a third recirculation zone, This is due to the thermal buoyancy effect and its importance can be characterized by the modified Froude number (Fr^*):

$$Fr^* = \frac{{u_s}^2}{gL\beta T_{super}} \tag{4.48}$$

where u_s is the local steel velocity magnitude, and g is the gravitational acceleration. L is a characteristic length, taken here as the hydraulic diameter of the strand $L = \frac{2t_s w_s}{t_s + w_s} = 0.53$ m.

The Fr^* indicates the relative strength of inertia to buoyancy and is shown in Figure 4.16 (see page 59) for the 30 K superheat case. In the jet and upper mold region, Fr^* is large, indicating that inertia is more than ten times larger than buoyancy, so thermal buoyancy is expected to have a negligible effect there. This agrees with previous work [13]. However, deeper than 3 m below the meniscus, Fr^* drops to 0.1, indicating that the thermal buoyancy becomes more than ten times more important than inertia, and is expected to change the flow pattern in the the lower strand.

The internal bubble capture is related to the strand flow. Bubbles smaller than PDAS can be entrapped anywhere once they touch the solidification front. Bubbles larger than PDAS are easier to be engulfed at two places in the lower strand region: (1) in the relatively stagnant region on the IR face, and (2) in the downward-flow region near the narrow faces, where the flow velocity is close to the bubble terminal rising velocity. Particle capture bands can form in these two general regions on both the IR and OR faces.

Figure 4.17 (a) - (d) (see page 59) shows the steel velocity on the width / 4 plane for both superheat cases. Both results show a lower recirculation zone in the thickness / length plane with a stagnation region near the IR face. Bubbles in this region tend to stay stationary in front of the solidification front and become engulfed. Comparing Figure 4.17 (b) and (d) (see page 59), higher (30 K) superheat shifts the stagnation region from ∼3 to ∼2.65 m below the meniscus, resulting in shallower bubble capture.

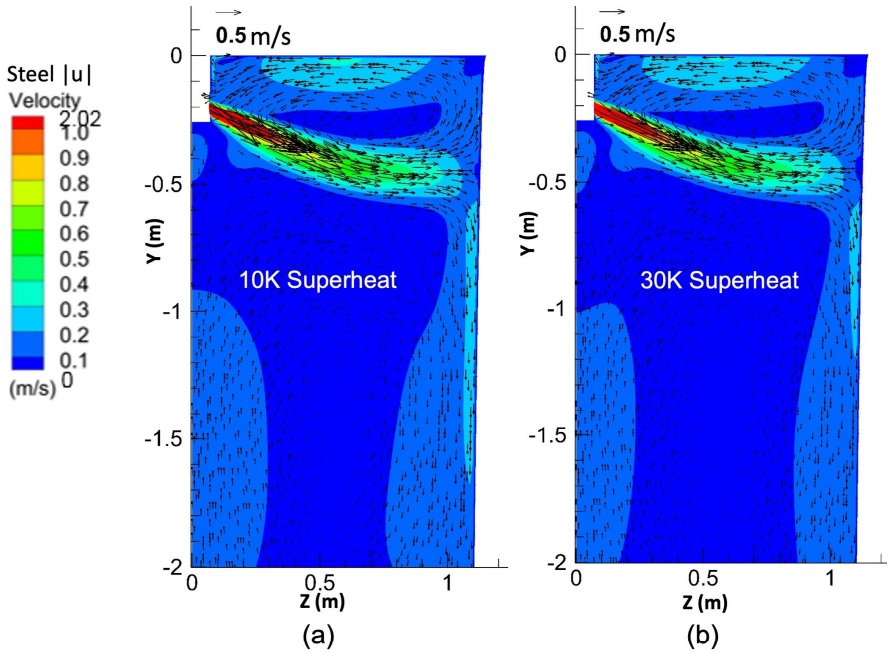


Figure 4.14 Close-up view of strand center-plane velocity: (a) 10 K superheat and (b) 30 K superheat case

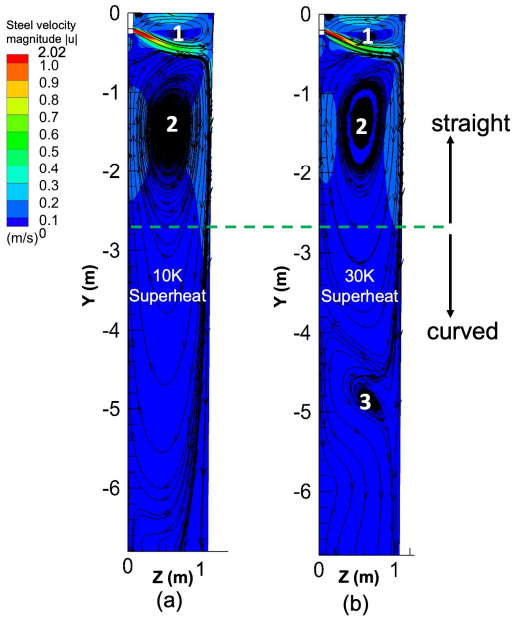


Figure 4.15 Strand center-plane velocity: (a) 10 K superheat and (b) 30 K superheat case

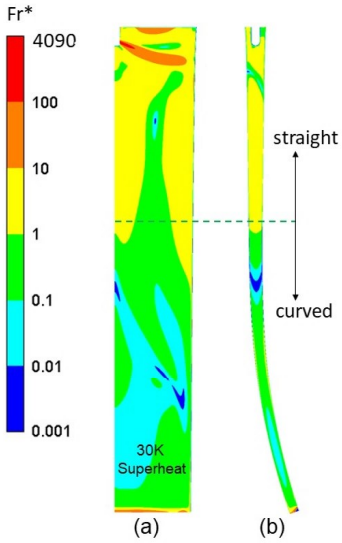


Figure 4.16 Modified Froude number distribution for 30 K superheat case: (a) On mold center plane and (b) On symmetry plane.

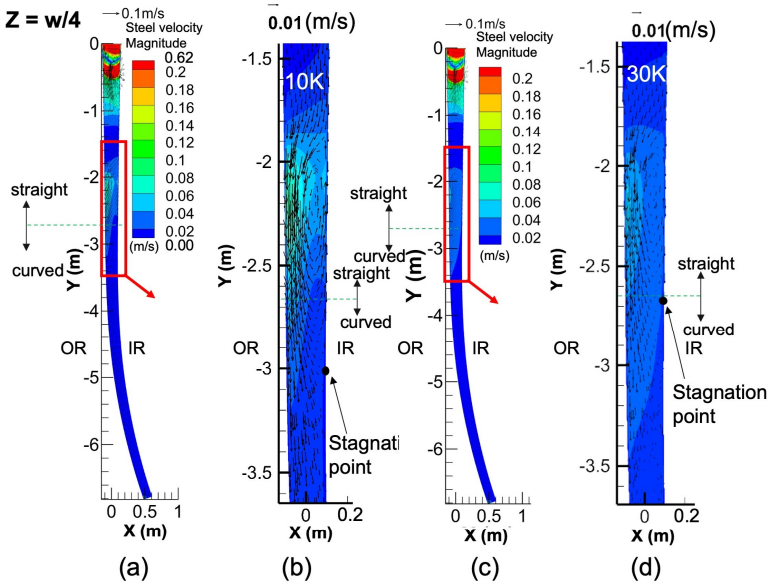


Figure 4.17 Flow pattern on width/4 plane: (a) 10 K superheat, (b) Close-up view, (c) 30 K superheat case, and (d) Close-up view

Figure 4.18 (a) - (d) shows the steel-velocity flow pattern on a thickness / length plane 0.15 m away from the NF. Figure 4.18 (b) shows that the velocity down the OR face is greater with lower (10 K) superheat, so large bubbles penetrate deeper into the strand, on the OR face than IR face, where their large terminal rising velocities are more likely to be overcome. More of these deep-penetrating bubbles are

eventually captured. The terminal velocity V_t can be calculated by [43]:

$$V_t = \sqrt{\frac{4}{3}\frac{\rho_s - \rho_{ar}}{\rho_s}\frac{d_p}{C_{D4}}g} \tag{4.49}$$

where ρ_g is the argon gas density, 0.43 kg/m^3. The drag coefficient C_{D4} can be calculated by $\frac{24}{Re}$ for bubbles smaller than 0.9 mm [43], and the Reynolds number Re is equal to $\frac{V_t d_p \rho_s}{\mu_s}$. The terminal velocity of $d_p = 0.6$ mm bubble is 0.218 m/s, which matches the velocity in the left top region near the OR face in Figure 4.18 (b), meaning easier capture of 0.6 mm bubbles inside that region.

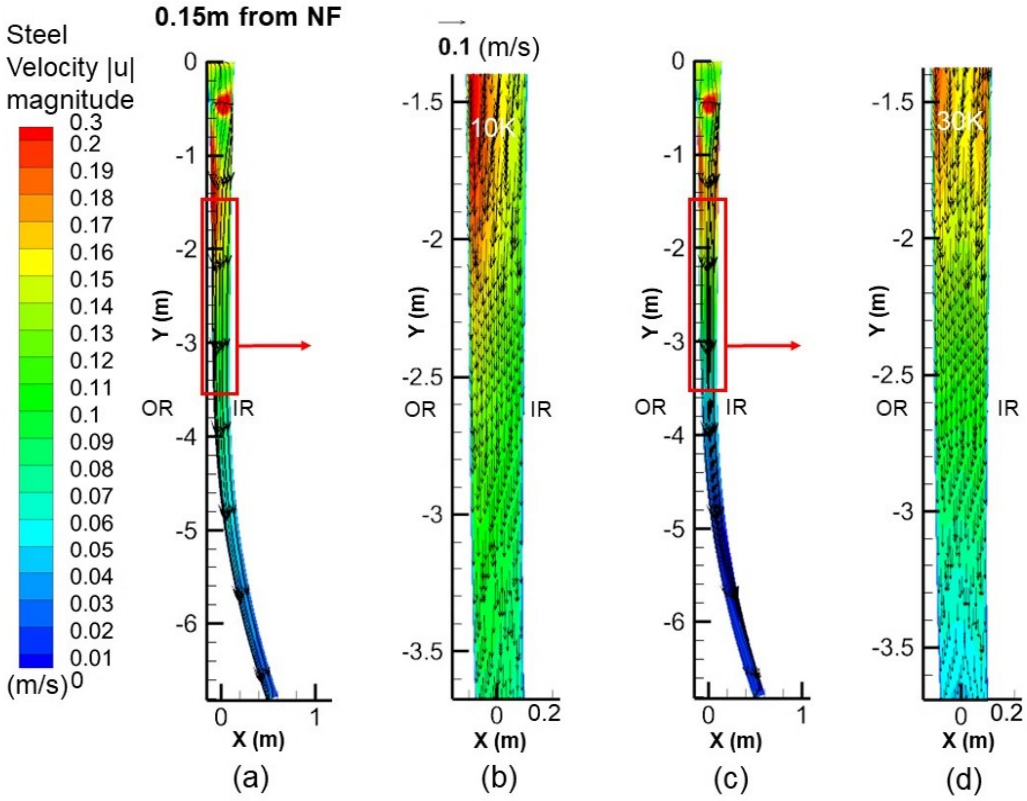


Figure 4.18 10 K superheat case: (a) Steel velocity magnitude and vector 0.15 m from NF, (b) Close-up view; 30 K superheat case: (c) Steel velocity magnitude and vector 0.15 m from NF, (d) Close-up view

Higher (30 K) superheat causes more uniform downward flow after hitting the NF, as a result of the lower third recirculation zone (∼4 m - ∼6 m below meniscus) pushing the velocity near the OR face up. This means that relatively more bubbles will flow downward along the IR face than OR face with higher (30 K) superheat.

4.12 Temperature distribution Results

Figure 4.19 shows the temperature distributions in the strand center plane and a horizontal plane 5 mm below the meniscus for the 10 and 30 K superheat cases. For both superheat cases, the molten steel temperature is generally higher near the jet and upper recirculation zone, and gradually cools down as the steel flows for a longer time and is consequently transported deeper down the strand. These results mainly show the superheat temperature distribution, as the temperature contours generally have ranges of 10 and 30 K for the two cases. An important exception occurs near the top surface for the 10 K case. For this case, within 5 mm of the top surface, Figure 4.19 (c) shows that the liquid temperature drops below the solidification temperature. This may form a frozen-steel crust or "island", which may become captured into the solidified shell, together with its attached slag, bubbles, and inclusions, and eventually form a giant cluster of internal defects in the final product. With 30 K superheat, the temperature everywhere is high enough to keep the steel liquid, except near the meniscus, where the two-dimensional heat losses into both the solidification front and top slag layer enable meniscus hooks to form.

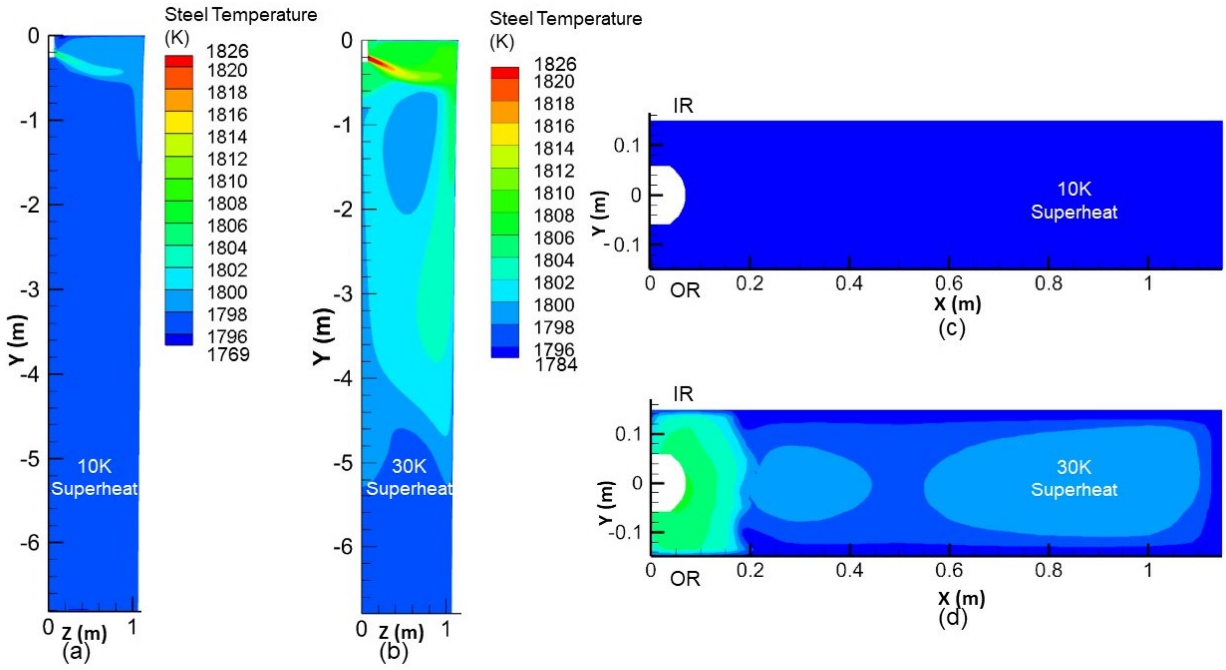


Figure 4.19 Temperature distribution on center plane across slab width for (a) 10 K superheat case and (b) 30 K superheat case; Temperature distribution near the top surface for (c) 10 K superheat case and (d) 30 K superheat case

In some steel samples, the UT measurements reveal a giant internal defect, which consists of a very large cluster of particles. For example, Figure 4.20 (a) shows such a defect, measuring 4.9 × 587 × 1907

mm (thickness × width × length) in the plate, which corresponds to 70 × 345 × 227 mm in the slab before rolling. Plant observations show this type of defect can occur with both low (10 K) and high superheat (30 K) but is more common with lower (10 K) superheat. This is consistent with the capture of a large frozen island / crust of the top surface, predicted in the simulations, after being broken up by rolling. Such large internal cluster defects have been observed in previous work [21, 78].

4.13 Superheat Dissipation Results

For the low (10 K) superheat case, 170 kW of superheat enters the half domain through the nozzle port. The top surface slag layer is predicted to remove 57 kW (32%) heat, as specified in Chapter 3. Much of the superheat is removed by the mold walls, including 21% (WF) and 7% (NF). The fraction removed to the mold NF is more than double its area fraction, owing to the high gradients caused by jet impingement. Below the mold, another 21% is removed from the straight strand and 17% from the curved strand. Less than 2% of the superheat exits the domain outlet by advection.

For the high (30 K) superheat case, results are generally similar, with 35%, 8%, 26%, and 20% of the 505 kW of superheat entering the half domain being removed by the mold WF, mold NF, straight strand below the mold, and curved strand respectively. Only 0.5% of the superheat exits the domain. The top slag layers remove 57 kW heat again. However, this represents only 10% of the total superheat entering the system. This calculation is consistent with previous work [12], where only 4% of the superheat was predicted to be removed by the top surface, for a case with 57 K superheat.

4.14 Particle Capture Model Validation

UT tests have been conducted on sixteen plates, measuring 0.21 × 3.9 × 36.2 m (thickness × width × length) rolled from 0.3 × 2.3 × 4.31 m slabs samples. Figure 4.20 shows an example of the UT tests for a case with 29 K superheat.

Figure 4.20 (a) shows an end view of the measured particle capture locations, converted back to the original slab dimensions. The black dots are extracted from the UT map by an online tool WebPlotDigitizer. Most of the captured particles are located around the transition line from vertical to curved (the green dashed line) on the IR face, forming a capture band (between the two red dashed lines) especially on the inside radius. This capture band is due to the stagnation in this region shown in Figure 4.20 (b), as discussed previously. It is located 40 - 60 mm below the strand surface, which corresponds to ∼1.6 - 3.5 m below meniscus along the strand, as shown in Figure 4.20 (b), This is similar to the location of capture bands observed below the surface in previous work: 50 - 65 mm in [26] and 35 - 45 mm in [25].

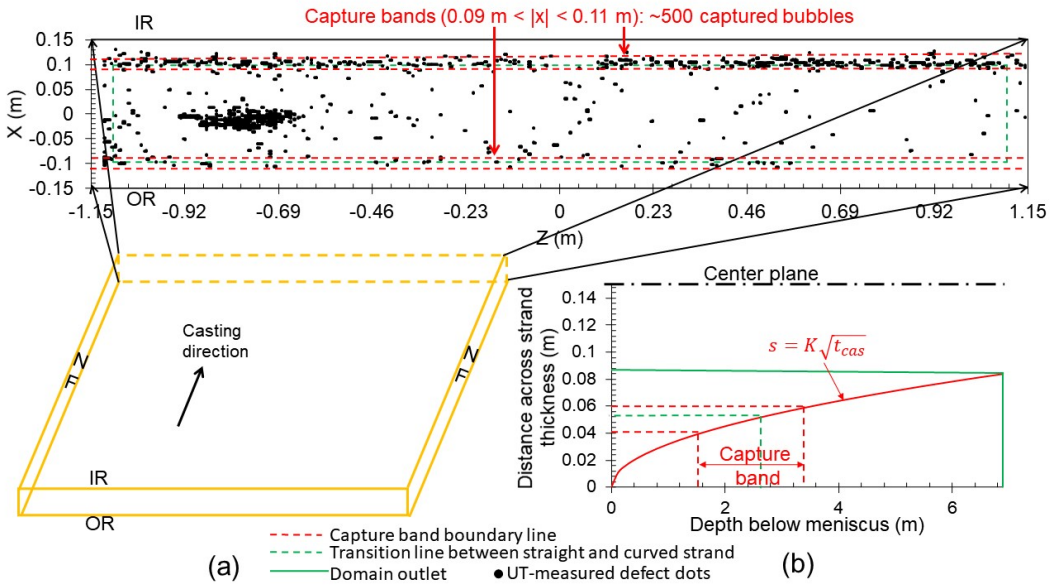


Figure 4.20 UT measurements (a) Schematic and particle capture band location and (b) Capture band location on the shell thickness evolution profile

Figure 4.21 (see page 64) shows results for a different slab. The red dots show the calculated capture locations of 0.6 mm bubbles with 10 K (Figure 4.21 (a) (see page 64) and 30 K superheat (Figure 4.21 (b) (see page 64)). The latter results are compared with another UT map (black dots), and reveal 202 particles in the capture band. Only the capture locations of large ($d_p > 0.5$ mm) bubbles are compared with the UT results, because the latter cannot detect any small defects. Near to the meniscus, the overprediction is likely caused by the difficulty of detecting the internal particles near to the plate surface by the UT device. The simulation predicts more bubbles captured with lower (10 K) superheat, especially inside the IR and OR capture-band regions (between the upper and lower pair of red dashed lines), due to the deeper stagnation region near IR face and higher downward velocity near OR face.

In Figure 4.21 (b) (see page 64), far fewer bubble are predicted in the capture-band region than measured. This is because only 30,000 of the 0.6 mm bubbles are injected, which is far less than the actual number of 0.6 mm bubbles in the strand during the same sample casting time, as explained in section 4.7.

Equation 4.45 can be used to estimate the true number of bubbles captured inside the capture band of the sample, N_{ci}, by setting $i = 0.6$ mm. These post-processed results for both 10 K and 30 K simulations are compared with the 16 UT measurements of rolled slab samples in Figure 4.22 (see page 64). The predictions are now quantitatively consistent with the measurements. The higher superheat case shows fewer defects in both IR and OR, which agrees with the plant measurements. Both the simulation and measurements show that the number of bubbles captured inside the capture band gradually decreases with

increasing superheat. With higher superheat, more bubbles are captured on the IR face than on the OR face. With lower superheat, the trend is reversed as more bubbles are captured on the OR face. These results which are observed in both simulation and measurements, can be explained by the flow pattern in Figure 4.18, as discussed previously.

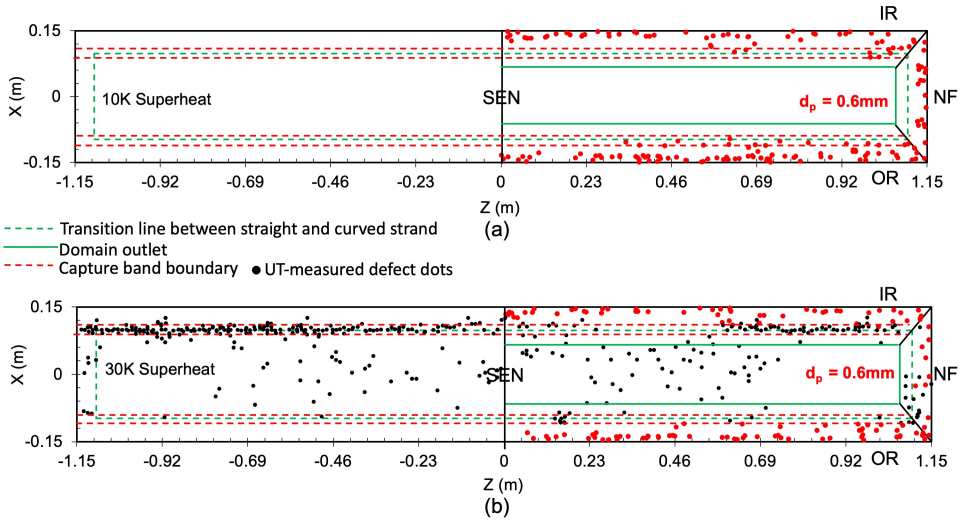


Figure 4.21 End view – (a) 10 K superheat case: Capture locations of 0.6 mm bubbles and (b) 30 K superheat case: comparison between 0.6mm bubble capture location and UT measurement

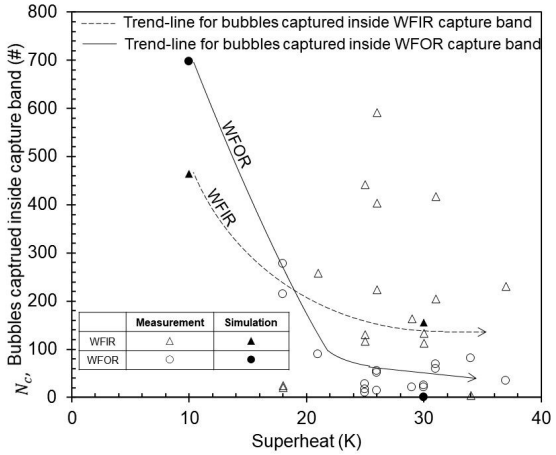


Figure 4.22 Model validation –number of bubbles captured inside capture band from simulation vs. from UT maps

4.15 Particle Capture Results

Figure 4.23 (see page 66) shows the end view of capture locations for different bubbles with 10 K and 30 K superheat. In addition to many particles captured near the strand surface, the model predicts a capture band on both the IR and OR faces, near the transition lines between the straight and curved parts of the strand. These capture bands are due to bubbles being entrapped while moving down near the NF into the lower recirculation zone. This leaves a thin, relatively clean space between the surface region and the capture band.

Figure 4.24 (see page 66) shows the escape location of different bubbles into top slag layer with 10 K and 30 K superheat. Small bubbles ($d_p < 0.5$ mm) tend to float up and escape uniformly over the top surface area. Large bubbles ($d_p > 0.5$ mm) tend to escape near the SEN, due to their large buoyancy floating them up immediately after exiting the nozzle port.

Figure 4.25 (see page 67) and Figure 4.26 (see page 67) show the capture locations for different bubbles on wide-face inner radius (WFIR) for 10 K superheat case and 30 K superheat case. For both cases, smaller bubbles ($d_p < 0.5$ mm) flow deeper down the strand and are mostly captured, because they tend to flow with the molten steel and be transported deeper. Large bubbles ($d_p > 0.5$ mm) tend to flow up into the slag layer, and very few of them are captured. For 0.3 mm bubbles, a sudden increase of capture occurs at the distance down the caster where the PDAS size increases to the bubble diameter. With higher (30 K) superheat, bubble capture locations are shallower due to larger thermal buoyancy, which lessens particle penetration depth. The results here suggest that 6 and 2 bubbles are captured inside the measured capture-band region for 10 K and 30 K superheat cases, respectively.

Figure 4.27 (see page 68) and Figure 4.28 (see page 68) show the capture locations for different bubbles on wide-face out radius (WFOR) for 10 K superheat case and 30 K superheat case. Trends are similar to the IR in Figure 4.25 and Figure 4.26. The lower (10 K) superheat case leads to more capture of 0.6 mm bubbles on the OR face inside the measured capture band, while the higher (30 K) superheat case shows the opposite trend. This is due to the flow pattern differences, shown in Figure 4.18 (b). The higher downward velocity near the OR face with lower (10 K) superheat transports more 0.6 mm bubbles deeper down the face and balances their terminal velocities.

Figure 4.29 (see page 69) and Figure 4.30 (see page 69) show the capture locations for different bubbles on NF for 10 K superheat case and 30 K superheat case. Higher superheat (30 K) exhibits shallower capture for all bubble sizes.

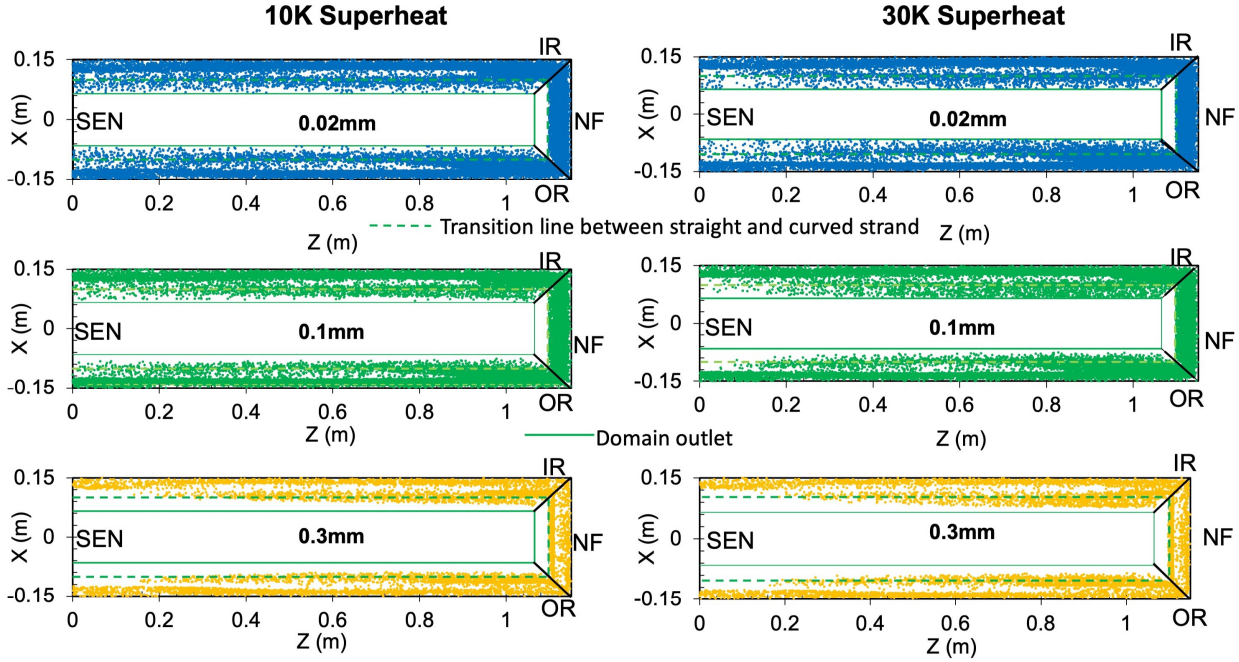


Figure 4.23 End view – Capture locations for different bubbles from 10 K superheat case and 30 K superheat case

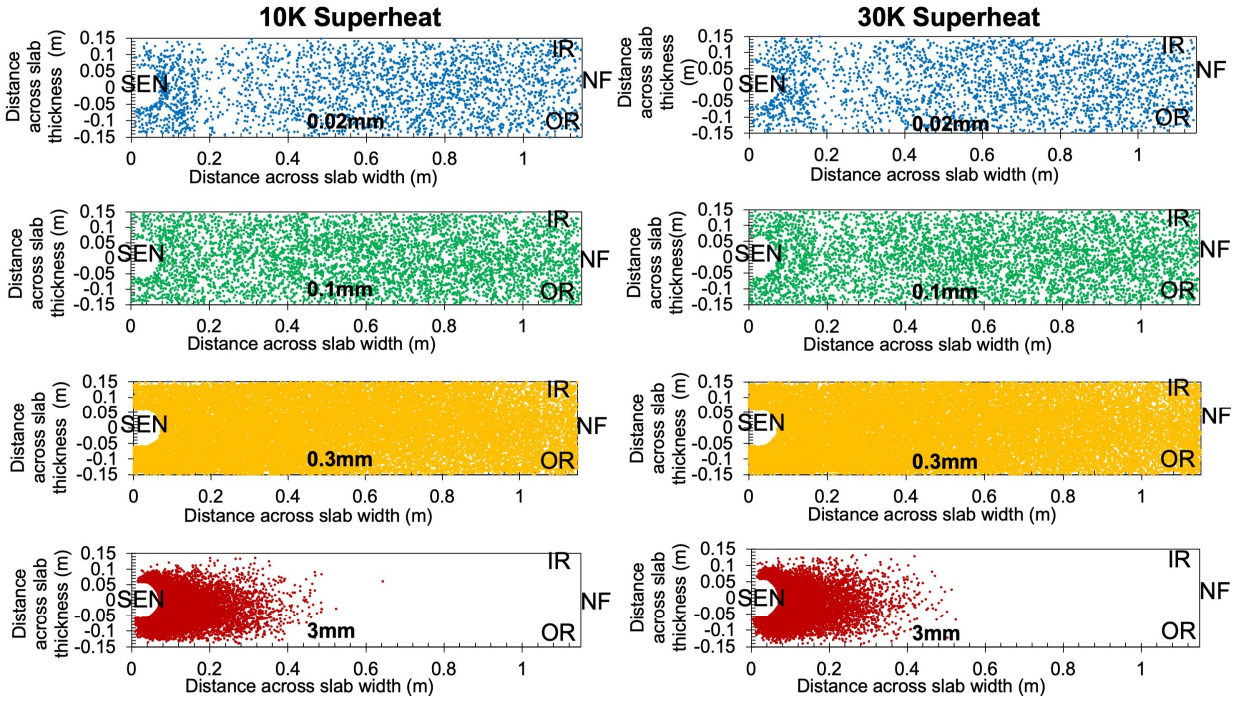


Figure 4.24 Top view – Escape locations into slag layer for different bubbles from 10 K superheat case and 30 K superheat case

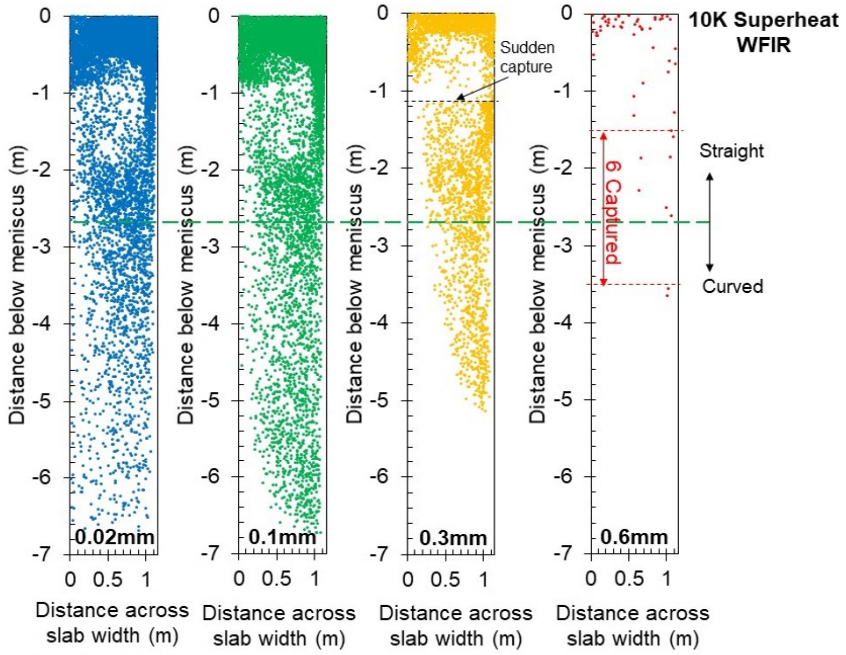


Figure 4.25 Front view – Capture locations on WFIR for different bubbles (10 K superheat case)

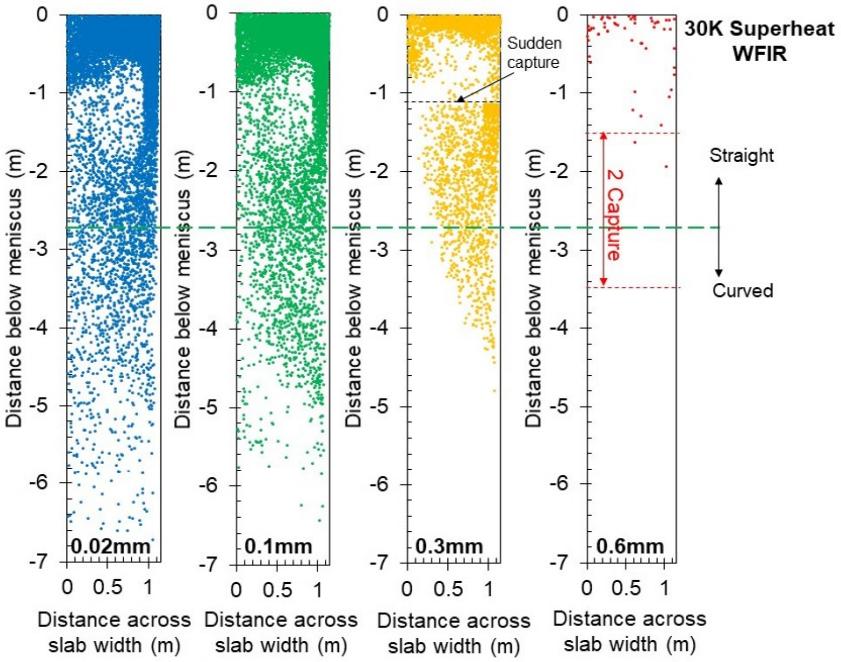


Figure 4.26 Front view – Capture locations on WFIR for different bubbles (30 K superheat case)

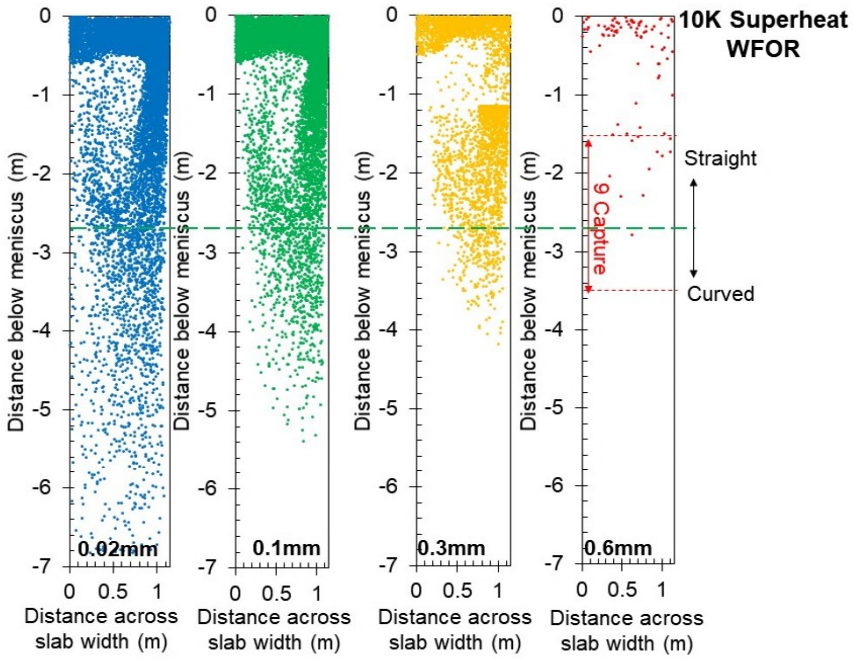


Figure 4.27 Front view – Capture locations on WFOR for different bubbles (10 K superheat case)

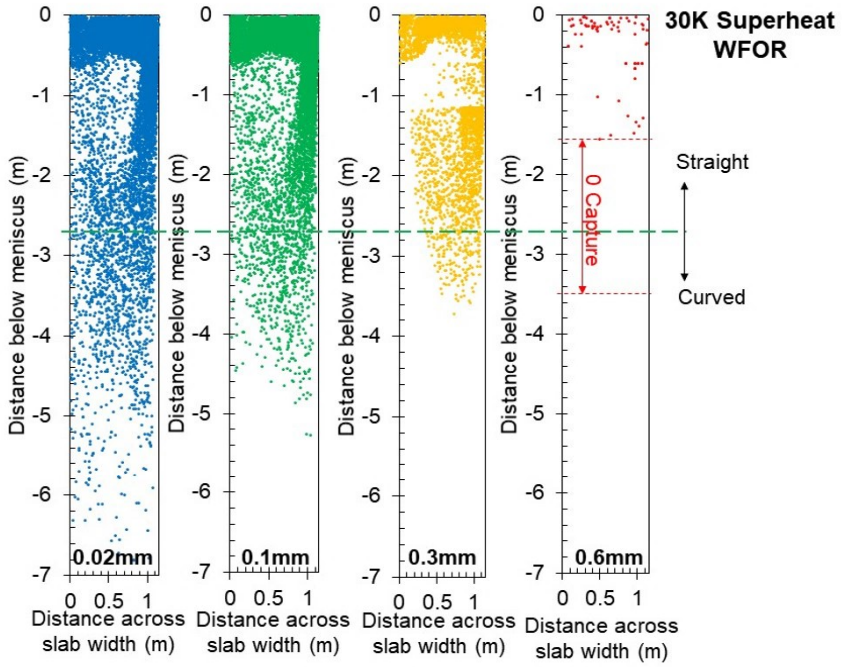


Figure 4.28 Front view – Capture locations on WFOR for different bubbles (30 K superheat case)

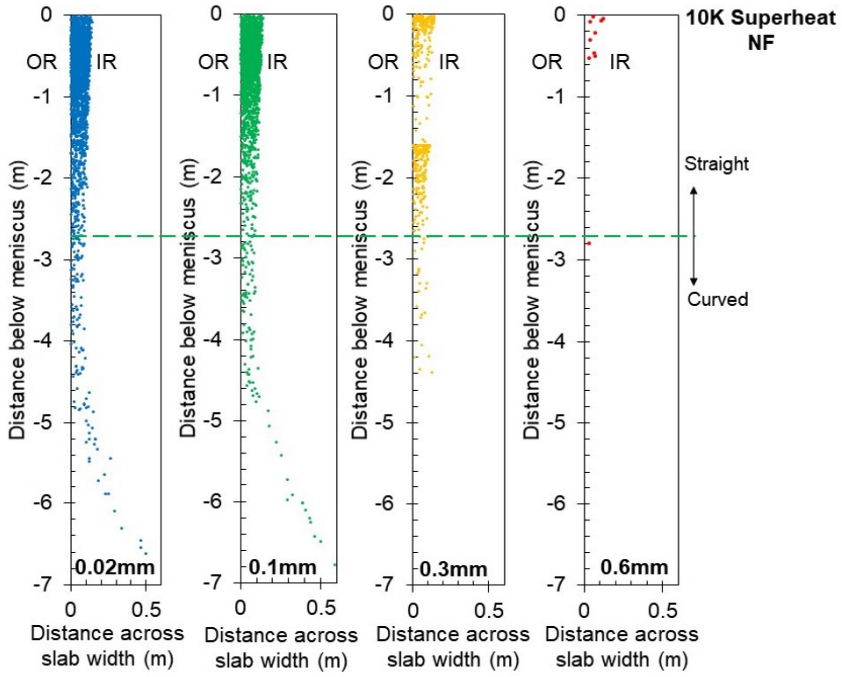


Figure 4.29 Side view – Capture locations on NF for different bubbles (10 K superheat case)

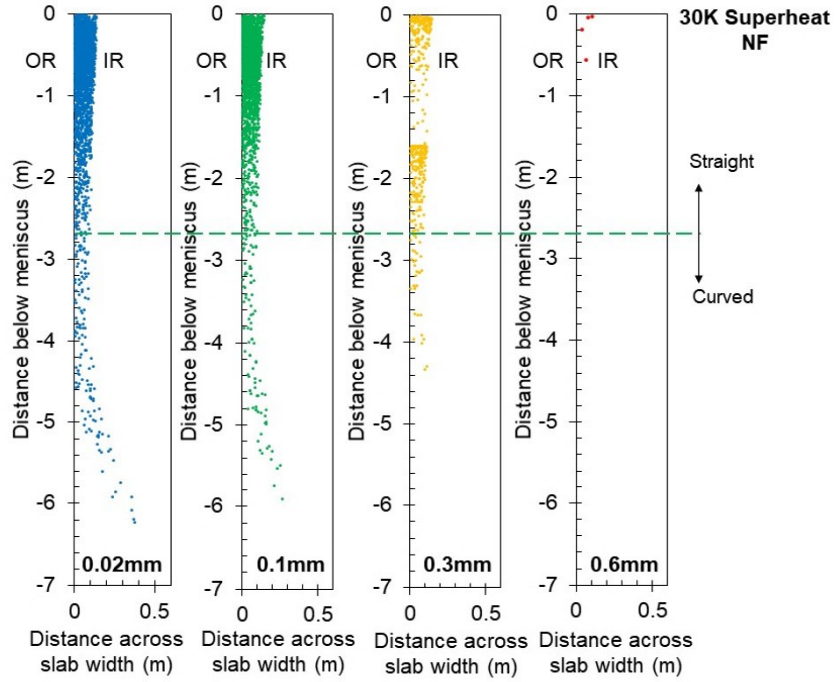


Figure 4.30 Side view – Capture locations on NF for different bubbles (30 K superheat case)

Figure 4.31 and Figure 4.32 show the capture and escape fractions by different regions of the strand for both superheat cases. The escape fraction into the top slag layer increases with increasing bubble diameter. Only ∼9% of 0.02 mm bubbles are removed by the slag layer, while almost all 0.6 mm bubbles float up and escape into the top slag layer. Higher (30 K) superheat has a slightly higher escape fraction.

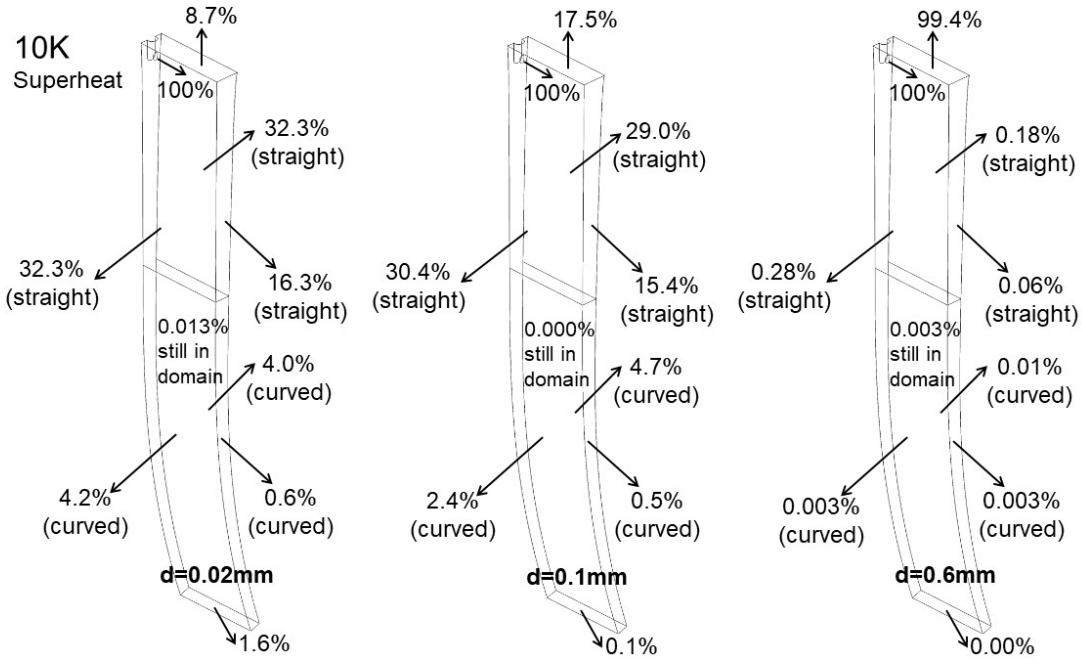


Figure 4.31 10 K superheat case – capture and escape fraction of different bubbles

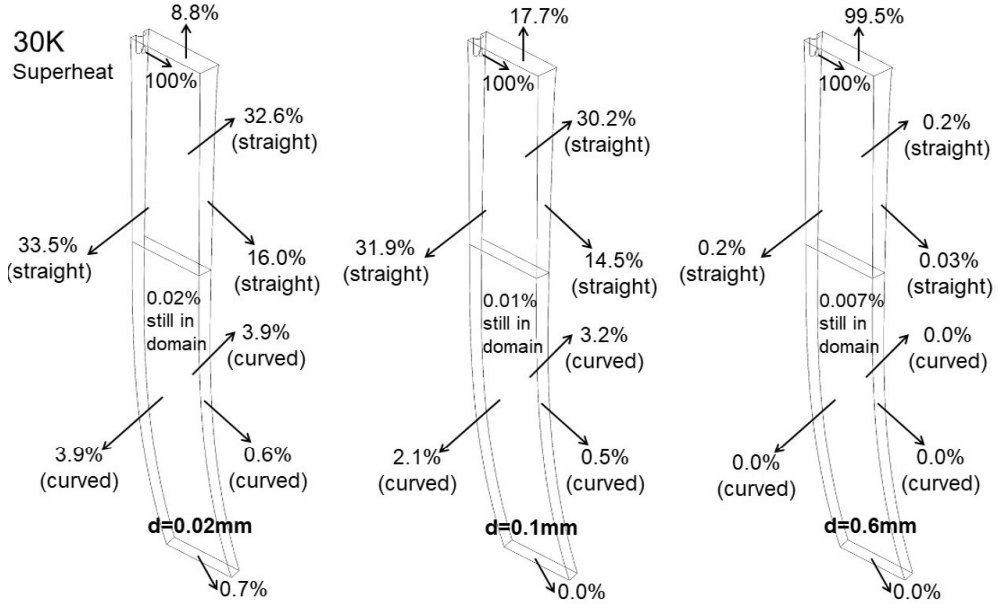


Figure 4.32 30 K superheat case – capture and escape fraction of different bubbles

In the hook zone, the lower (10 K) superheat case has higher capture by all three mechanisms (meniscus hooks, entrapment, and engulfment) relative to the 30 K case. For small bubbles ($d_p < 0.5$ mm), hook capture is negligible compared with the entrapment and engulfment mechanisms. The hook capture mechanism becomes increasingly important with increasing bubble size. For very small bubbles, ($d_p < 0.06$ mm), entrapment is the dominant capture mechanism everywhere, including inside the hook zone. Engulfment grows in importance with increasing bubble size. For all bubble sizes, capture in the hook zone is greater with the lower (10 K) superheat case. Several (16) large bubbles ($d_p > 0.5$ mm) are captured by the hook mechanism with lower (10 K) superheat, due to the deeper meniscus hooks. This includes a single 1 mm bubble. With the high superheat case, no large bubbles ($d_p > 0.5$ mm) are captured in the hook zone.

In the straight strand, including the mold, over 70% of 0.02 and 0.1 mm bubbles are captured by the shell (entrapment/engulfment). Only ∼10% of these small bubbles are captured by the curved part. For 0.6 mm bubbles, ∼0.016% are captured by the curved part with the 10 K superheat, but no large bubbles ($d_p > 0.5$ mm) are captured in this region with the higher (30 K) superheat. As expected, and observed in the plant, more bubbles are captured on the IR in the curved part of the strand.

Figure 4.33 shows the capture locations of different bubble sizes inside the hook zone on WFIR, WFOR, and NF, for 10 K superheat case and 30 K superheat case. It is apparent that more bubbles are captured by the hook zone with lower superheat due to deeper hook depth. No 0.6 mm and 1 mm bubbles are capture by the case with higher (30 K) superheat due to shorter hook.

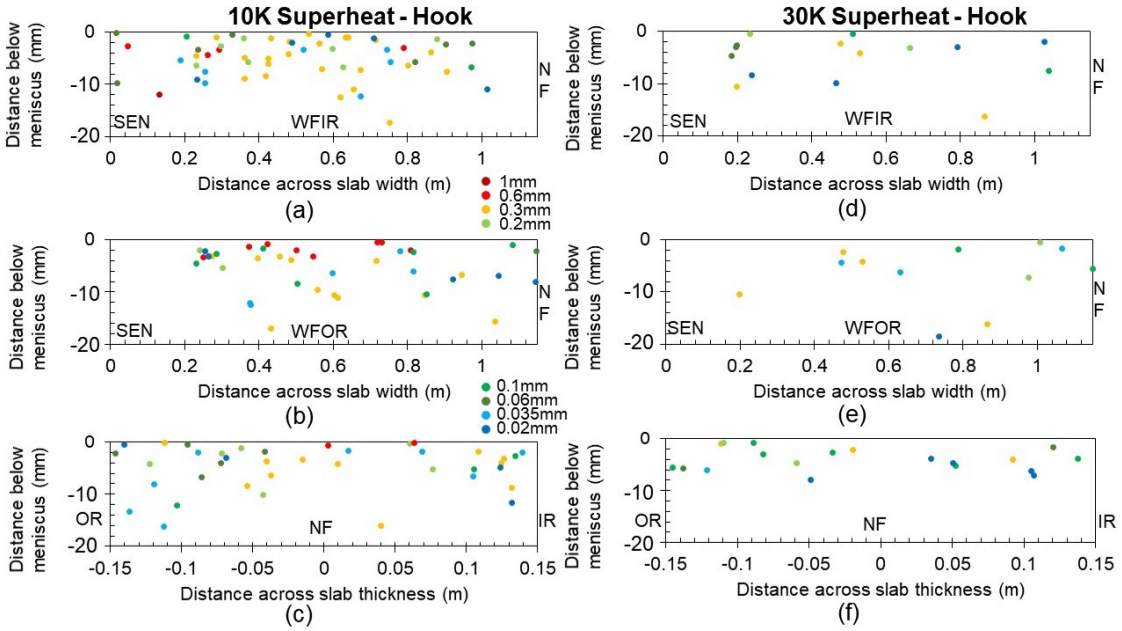


Figure 4.33 Capture locations for bubbles by hook capture mechanism for 10 K superheat case on (a) WFIR, (b) WFOR, and (c) NF; for 30 K superheat case on (d) WFIR, (e) WFOR, and (f) NF

Figure 4.34 compares the overall capture fraction of different bubbles with 10 and 30 K superheat using Equation (4.46), and their capture fractions by the hook, entrapment, and engulfment mechanisms.

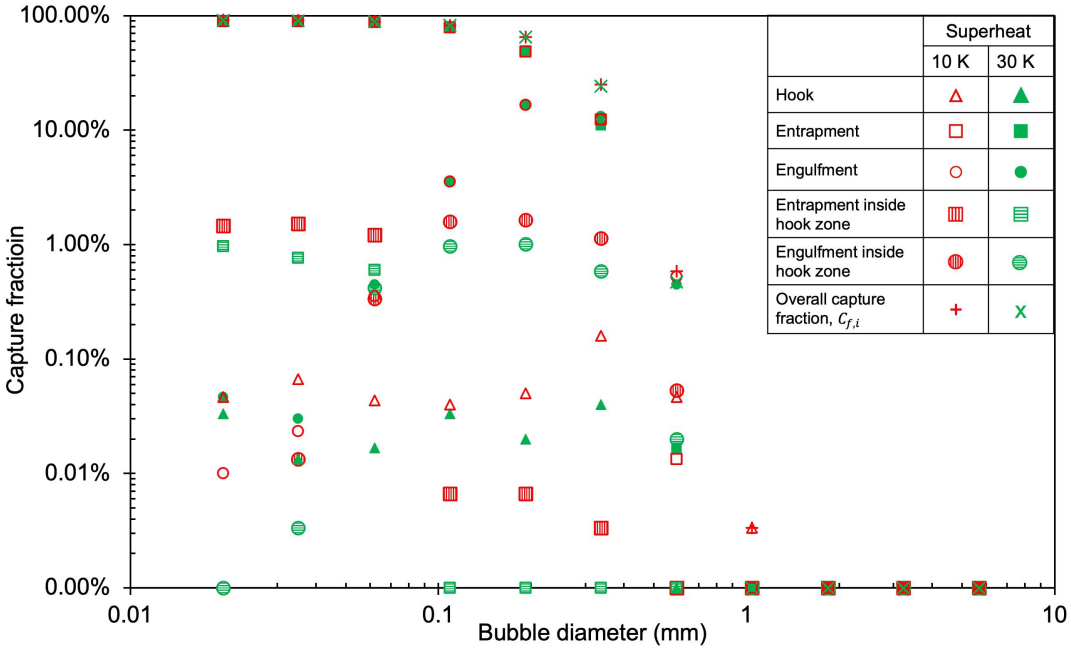


Figure 4.34 Overall capture fraction of different bubbles, and breakdown by hook, entrapment, and engulfment mechanisms.

For small bubbles ($d_p < 0.5$ mm), the overall capture fraction is very similar for both superheat cases. For $d_p < 0.1$ mm bubbles, the overall capture fraction reaches ~90% and gradually declines with increasing bubble sizes. For large bubbles ($d_p > 0.5$ mm), the higher (30 K) superheat case has slightly less overall capture (0.05% vs. 0.06%). The single bubble captured that was larger than 1 mm was by the hook capture mechanism with 10 K superheat, and represents 0.003% of those injected.

All of the other $d_p > 1$ mm bubbles escape into the top slag layer for both superheat cases. This agrees with previous work that almost all large bubbles escape to the top surface [24, 27, 80]. For both superheat cases, entrapment dominates over engulfment for $d_p < 0.3$ mm bubbles, because these bubbles are smaller than PDAS and can easily flow in between two primary dendrite arms to be captured whenever they touch the solidification front. For $d_p > 0.3$ mm bubbles, engulfment naturally becomes more important.

Figure 4.35 compares the capture rates in the slab samples for the 10 and 30 K superheat cases using Equation (4.47), where the overall capture fraction of bubbles with diameter d_{pi}, $C_{f,i}$, is shown in Figure 4.34.

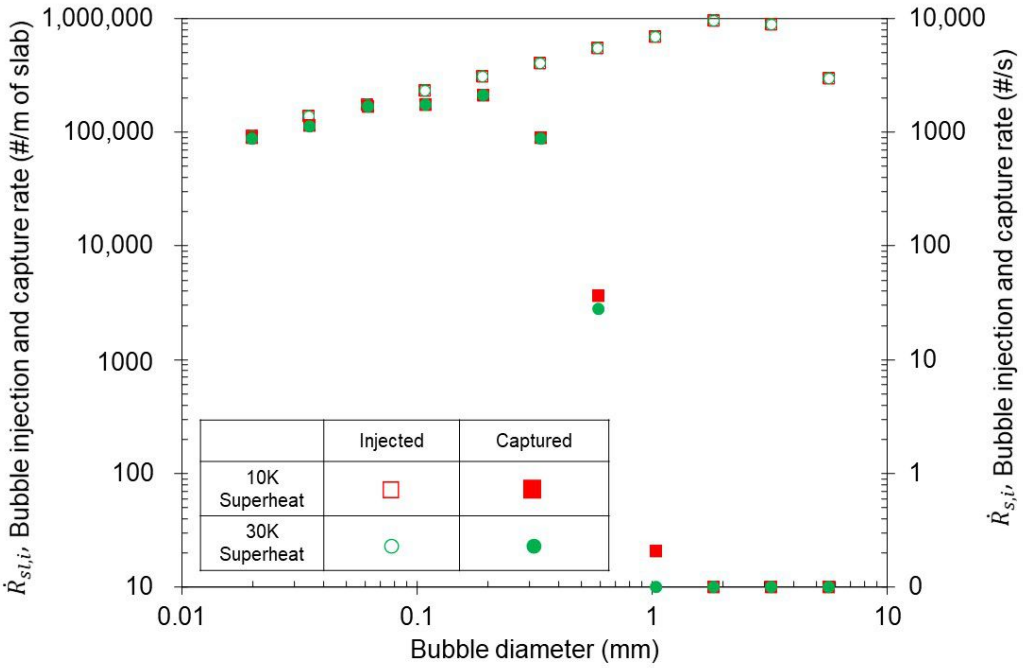


Figure 4.35 Injection and capture rates for all bubble sizes.

There are about eleven 1 mm bubbles captured per meter of the slab with 10 K superheat, while no bubbles larger than 1 mm are captured with higher (30 K) superheat.

The new findings presented here show the important drawbacks of low superheat on meniscus solidification and fluid flow deep in the caster, and the corresponding effects on increasing particle-capture defects. This work neglects the important effects that the superheat has on fluid flow and the inclusion of particle distribution in the tundish. These differences in tundish flow behavior serve to greatly augment the effects of superheat in the strand, which are identified here.

4.16 Conclusion

A new modeling approach was developed to investigate the multiphase turbulent flow, superheat dissipation, particle transport, and capture during steady continuous casting of steel slabs. The predictions were validated with plant measurements including nail board dipping tests of surface velocity and UT measurements of particle capture locations. Good numerical convergence (to within 10^{-4} for the nozzle flow and 10^{-7} for the strand flow) was achieved using a half nozzle and strand domain with structured hexahedral cells aligned with the jet direction.

Findings regarding the fluid flow include:

- Superheat has a negligible effect on the flow in the upper strand region where inertia is dominant over thermal buoyancy.
- Superheat has a significant effect in the lower strand, especially > 3 m below the meniscus where thermal buoyancy dominates the flow and creates multiple complicated recirculation regions.
- The flow pattern shows two locations for large bubbles to be engulfed: (1) near the stagnation point on the IR face, where the bubble tends to stay stationary, and (2) in the downward flow region near IR and OR faces, where the downward velocity balances the bubble terminal velocity. Capture bands can form in these two locations, which agrees with plant measurements.
- The lower (10 K) superheat case has a deeper stagnation region near the IR face and higher downward flow near OR face due to less thermal buoyancy, which allows steel to flow deeper and faster down the narrow faces below the jet impingement, leading to more and deeper capture.
- The lower (10 K) superheat case appears susceptible to a possible frozen crust or island near the top surface due to lower inlet temperature and heat loss into the slag layer. The capture of this island may bring slag, bubbles, and inclusions deep into the strand, forming a giant internal defect in the final product.
- The higher (30 K) superheat case leads to a more uniform downward velocity and a shallower stagnation region near the IR face, due to the lower third recirculation zone pushing the flow upwards.

The bubble transport and the capture results show:

- With the lower (10 K) superheat case, more 0.6 mm bubbles are captured on the OR face in the capture-band region than the IR face, due to the higher downward velocity transporting the bubbles deeper and balancing out their terminal velocities (0.218 m/s).
- The lower (10 K) superheat case leads to deeper meniscus hooks, which capture more particles, leading to more surface defects.
- The higher (30 K) superheat case leads to fewer and shallower capture of all bubble sizes as a result of stronger thermal buoyancy, indicating fewer internal defects.
- The higher (30 K) superheat case leads to relatively more 0.6 mm bubbles captured on the IR face than OR face in the capture-band region, due to a more uniform downward flow.
- Capture bands are predicted at $\sim$1/4 thickness from both the IR and OR strand surfaces, which matches UT maps.

- For small bubbles ($d_p < 0.5$ mm), clear capture bands are seen on both IR and OR faces near the transition line from vertical to curved parts of the strand from the end view. These two capture bands are due to bubbles being entrapped while moving down near the NF into the lower recirculation zone. This leaves a clean space.

- Small bubbles ($d_p < 0.5$ mm) tend to escape into the slag layer uniformly over its surface, while large bubbles ($d_p > 0.5$ mm) escape near the SEN due to the larger buoyancy, bringing them up immediately after exiting the nozzle port.

- The escape fraction into the top slag layer increases with increasing the bubble size due to the increasing buoyancy effect (∼9% to ∼100% for 0.02 and 0.6 mm bubbles, respectively).

- Superheat has little effect on the capture of bubbles smaller than 0.5 mm because they tend to flow with the steel, move down deeply, and become entrapped.

- Increasing superheat decreases the capture of 0.6 mm bubbles by 20% due to the complicated recirculation regions hindering the penetration of large bubbles deep into the caster

- Almost all very large bubbles ($d_p > 1$ mm) escaped into the top slag layer due to large buoyancy.

- For $d_p < 0.3$ mm bubbles, entrapment dominates over engulfment, because the bubbles are smaller than PDAS. As the bubble size increases > 0.3 mm, engulfment becomes increasingly important.

- Inside the hook zone: For small bubbles ($d_p < 0.5$ mm), the hook capture is negligible, relative to the entrapment and engulfment mechanisms. For very small bubbles ($d_p < 0.06$ mm), entrapment is the dominant mechanism, although engulfment grows in importance with the increasing bubble size. The single captured 1 mm bubble is by the hook capture mechanism with 10 K superheat.

4.17 Supplement Notes

Author contributions: Conceptualization, methodology, software, formal analysis, investigation, validation, visualization, writing—original draft preparation, M.L.; methodology, software, supervision, S.-M.C.; Data curation, resources, investigation, X.R.; conceptualization, methodology, formal analysis, investigation, resources, supervision, writing—review and editing, project administration, funding acquisition, B.G.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Science Foundation (grant no. CMMI 18-08731), the Continuous Casting Center, Colorado School of Mines, and Baosteel, Shanghai, P.R. China (grant no. 470108).

Acknowledgment: The authors thank Baosteel, Shanghai, P.R. China, for providing the casting conditions and plant measurements. Provision of Fluent licenses through the Ansys Inc. academic partnership program is much appreciated.

Conflict of interests: The authors declare no conflict of interest.

CHAPTER 5

LARGE EDDY SIMULATION FOR MULTIPHASE FLOW IN A STRAIGHT AND BENDING STRAND

5.1 Introduction

Large Eddy Simulation (LES) is more accurate than RANS modeling, because the large eddies which contain most of the turbulent energy are directly solved by LES while they are simply modelled with RANS. Besides, the LES turbulent model has been proved to give less and more reasonable capture of large bubbles than RANS model, due to the use of RWM in the RANS model which may overpredict the chance of a bubble hitting the wall [27].

The highly turbulent flow is always transient in the nozzle and strand, which affects the capture locations of detrimental particles. So, there is a great incentive to investigate this phenomenon with the more accurate LES model.

5.2 Computational Models and Solution Procedure

In this chapter, a three-dimensional finite-volume LES-DPM-coupled model is adopted to predict the transient flow of molten steel and argon gas in the nozzle and strand. A converged steady-state single-phase iso-thermal RANS velocity and turbulence field on a half-nozzle and half-mold domain is mirrored about its symmetry plane and interpolated onto a full-domain mesh as the initial condition of the LES simulation at t=0s. To stabilize the simulation, iso-thermal condition is employed until 20.56s, then the energy balance equation is solved with 30K superheat at the inlet ($T_{in} = 1826K$). The steel density is temperature-dependent and is calculated by Equation 4.15 The commercial software ANSYS FLUENT is used for these simulations.

The shell thickness and particle capture mechanisms are the same as those in chapter 4. Two-way-coupled argon bubbles are injected every time step with the same Rosin-Rammer size distribution, shown in Figure 4.2.

5.2.1 Turbulence Model - Large Eddy Simulation

The transient mass, momentum and energy balance equations with the single-phase LES model are solved as follows:

$$\frac{\partial \rho_s}{\partial t} + \frac{\partial}{\partial x_i}\rho_s u_{is} = S_{s-mass-sink} \tag{5.1}$$

$$\frac{\partial(\rho_s u_{is})}{\partial t} + \rho_s u_{js}\frac{\partial u_{is}}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}[(\mu_s + \mu_{sgs})(\frac{\partial u_{is}}{\partial x_j} + \frac{\partial u_{js}}{\partial x_i})] + F_{Bi,t} + S_{is-momentum-sink} + S_{DPM} \tag{5.2}$$

$$\frac{\partial(\rho_s h_s)}{\partial t} + \frac{\partial \rho_s u_{is} h_s}{\partial x_i} = \frac{\partial}{\partial x_i} k_{eff}\frac{\partial T_s}{\partial x_i} + S_{hs} \tag{5.3}$$

where μ_s and μ_{sgs} are the molecular viscosity of molten steel and subgrid-scale turbulent viscosity. S_{DPM} is the momentum source term from the motion of discrete particles. The mass and momentum sink terms can be calculated in the same way as in Chapter 4. The energy balance term can be calculated based the mass sink term:

$$S_{hs} = S_{s-mass-sink} c_{ps}(T_{local,s} - T_{ref}) \tag{5.4}$$

where T_{ref} is reference temperature, 298.15K.

k_{eff} is the effective thermal conductivity and is equal to $k_s + k_{sgs}$. k_s is the steel thermal conductivity and k_{sgs} is the turbulent thermal conductivity, which is calculated by the following equation:

$$k_{sgs} = \frac{c_p \mu_{sgs}}{Pr_t} \tag{5.5}$$

where Pr_t is the turbulent Prandtl number, 0.85.

The Smagorinsky-Lilly model [118] is used to calculate the subgrid-scale turbulent viscosity, mu_{sgs}. The equation is as follows:

$$\mu_{sgs} = \rho_s L_s^2 \|\bar{S}\| \tag{5.6}$$

where $\|\bar{S}\| = \sqrt{2\bar{S}_{ij}\bar{S}_{ij}}$, and $\bar{S}_{ij}$ is the rate-of-strain tensor for resolved scale. L_s is the mixing length for subgrid scales and can be calculated by $L_s = min(\kappa y, C_s \triangle)$, where κ is the von karman constant, y is the distance to the closest wall, C_s is the Smagorinsky constant, 0.1, and $\triangle$ is the cubic root of the cell volume, $V^{\frac{1}{3}}$.

5.2.2 Lagrangian DPM Model

The Lagrangian DPM Model is used to simulate argon bubbles transport in the molten steel pool in the caster in a two-way-coupled way with the LES model.

The following momentum balance equation is solved for each bubble with velocity u_{ip}:

$$\frac{Du_{ip}}{Dt} = \underbrace{\frac{18\mu}{\rho_p d_p^2}\frac{C_{D2}}{24}(u_{is}-u_{ip})\frac{\overbrace{\rho_s d_p \mid u_{ip}-u_{is} \mid}^{Re_p}}{\mu}}_{F_{iD}} + \underbrace{0.5\frac{\rho_s}{\rho_p}(\frac{Du_{is}}{Dt}-\frac{Du_{ip}}{Dt})}_{F_{iVM}} + \underbrace{\frac{\rho_s}{\rho_p}u_{ip}\frac{\partial u_{is}}{\partial x}}_{F_{iP}} + \underbrace{\frac{g(\rho_p-\rho_s)}{\rho_p}}_{F_{iB}} \tag{5.7}$$

where F_{iVM} is virtual mass force, per unit bubble mass an additional force required to accelerate the surrounding fluid when the bubble is accelerated. F_{iP} and F_{iB} are pressure gradient force and buoyancy/gravity force per unit bubble mass. Re_p is the particle Reynolds number. F_{iD} is the drag force per unit bubble mass, and the drag coefficient, C_{D2}, in DPM model was computed based on the drag law proposed by Moris *et.al*[98].

$$C_{D2} = a_1 + \frac{a_2}{Re_p} + \frac{a_3}{Re_p^2} \tag{5.8}$$

$$a_1, a_2, a_3 = \begin{cases} 0, 24, 0 & \\ 3.690, 22.73, 0.0903 & 0.1 < Re_p < 1 \\ 1.222, 29.1667, -3.8889 & 1 < Re_p < 10 \\ 0.6167, 46.50, -116.67 & 10 < Re_p < 100 \\ 0.3644, 98.33, -2778 & 100 < Re_p < 1000 \\ 0.357, 148.62, -47500 & 1000 < Re_p < 5000 \\ 0.46, -490.546, 578700 & 5000 < Re_p < 10000 \\ 0.5191, -1662.5, 5416700 & Re_p > 10000 \end{cases} \tag{5.9}$$

In this DPM model is coupled with LES model, because forces are mutual, S_{DPM} in Equation 5.2 is calculated as follows:

$$S_{DPM} = -(F_{iD} + F_{iVM} + F_{iP} + F_{iB})\dot{m}_p \triangle t \tag{5.10}$$

where $\dot{m}_p$ is the mas flow rate of argon gas and $\triangle t$ is the time step size for particle trajectory calculation, which, in this study, is the same time step size for LES model.

5.3 Flow Model Domain, Mesh, and Boundary Conditions

The flow model domain, including the complete slide gate, nozzle, and liquid pool in the strand, is shown in Figure 5.1 (a). The mesh is created in ANSYS Workbench and includes ∼8 million structured hexahedral cells. The Y^+ value ranges from 1 to 200. In most regions, the Y^+ is around 1, so the boundary layer is resolved. In regions near the slide gate and the jet, the Y^+ value is inevitably high, so the log-law wall function applies in these regions. Figure 5.1 (b) and (c) shows the center plane mesh in the nozzle and strand region, respectively. The mesh is refined near the slide gate due to high velocity gradient, and it is

built aligned with nozzle jet direction for 0.5m and then being parallel to the Z-axis to minimize false dissipation.

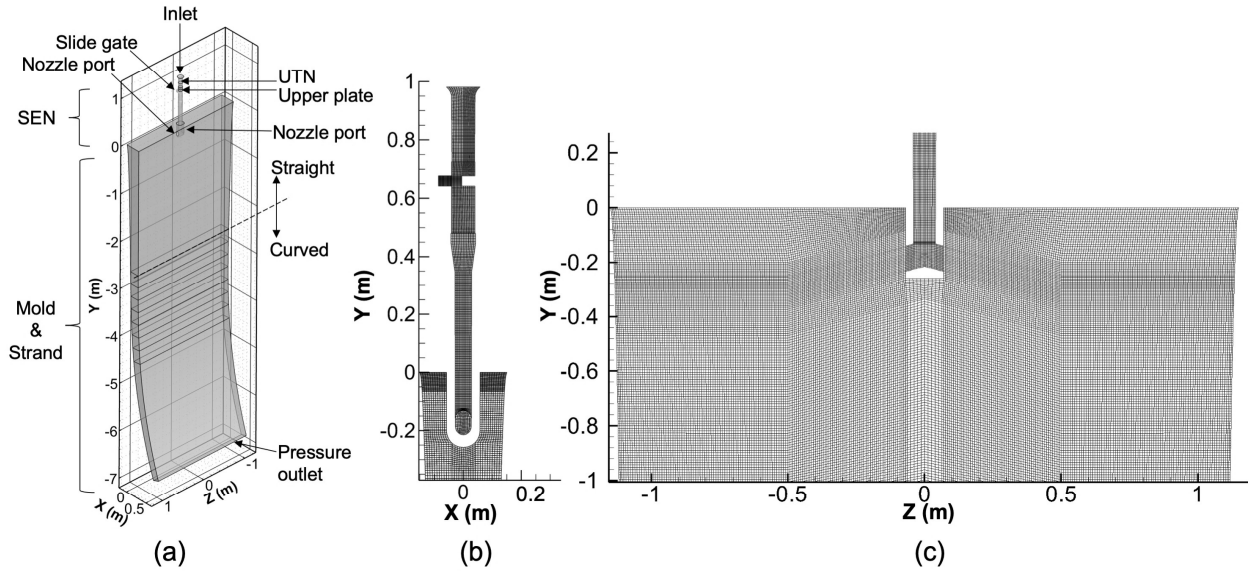


Figure 5.1 Geometry and mesh for (a) whole domain (b) nozzle center plane and (c) mold/strand center plane

Boundary conditions are shown in Figure 5.2. For the steel-phase flow simulation, constant mass flow rate 48.3kg/s is given at the inlet. Steel is entering in a direction normal to the inlet. Nozzle wall has 0.001m roughness and has non-slip boundary condition. The top surface is approximated as no-slip for the molten steel flow, due to the large viscosity of the slag layer. Other surfaces (wide and narrow faces) are also set as non-slip walls with mass, momentum, and energy sink UDFs on their adjacent cells (Appendix A). A pressure outlet condition is chosen over the nozzle outlet and the strand domain bottom, according to the ferrostatic pressure head of molten steel ($\rho_s gh$), where h is the distance below the meniscus. The outlet pressure is calculated by a UDF in Appendix A.

Heat transfer boundary conditions are shown in Figure 5.2 (b). The inlet temperature is with 30K superheat, 1826 K. Top surface has a heat flux into liquid slag layer, $164.67 kW/m^2$. Narrow and wide faces' temperatures are set at the steel liquidus temperature, 1796K, as it represents the solidification front. The reverse flow temperature is assumed to be 1796 K as well because the steel travels 7 m down to the strand and most of its superheat has been conducted into slag layer and steel shell. The steel density in the process is calculated based on its volumetric expansion coefficient β and equation 4.15.

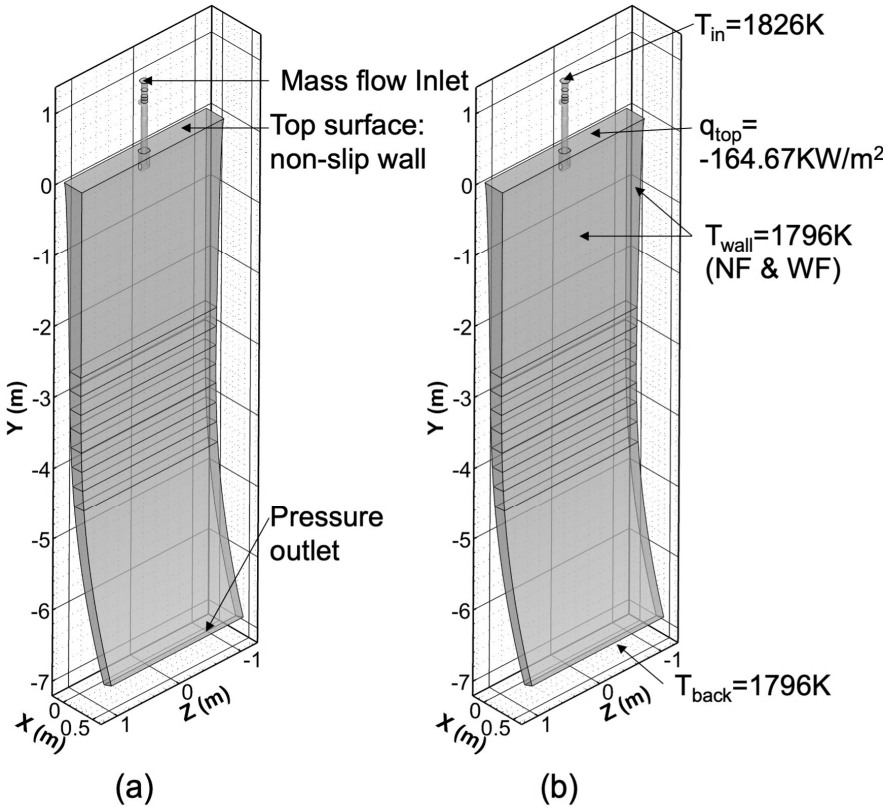


Figure 5.2 Boundary conditions for (a) Eulerian steel phase (b) Heat transfer, (c) Lagrangian bubble transport and capture simulations

A Semi-Implicit Pressure Linked Equations (SIMPLE) algorithm is used to solve the equation set of one mass balance, three momentum balance, and the LES equations with second-order upwind discretization scheme for convective terms. A single-phase iso-thermal RANS steel velocity field on a half domain is mirrored and interpolated onto the full-domain mesh as an initial condition at t=0s. The LES-DPM-coupled multiphase flow considered the interaction between steel and argon bubbles with a time step size 0.0025s. The temporal term is discretized with bounded second order implicit method. The transient iso-thermal flow is allowed to develop for 20.56s, then heat transfer is enabled for another 50.54s. The last 37.3s simulation is time-averaged for validation. For each time step, the residuals reach 10^{-5}. The simulation is conducted on the high-performance computer Mio with 96 computational nodes for $\sim$6 months.

5.4 Fluid Velocity Results

Figure 5.3 shows the initial condition at t=0s from the single-phase iso-thermal RANS simulation. A classic symmetrical double-roll flow pattern is seen. The nozzle jet hits the narrow face and gets split into

two streams, one goes upwards, forming a upper recirculation zone. The other stream goes down for ∼5.5m, forming a longer lower recirculation zone.

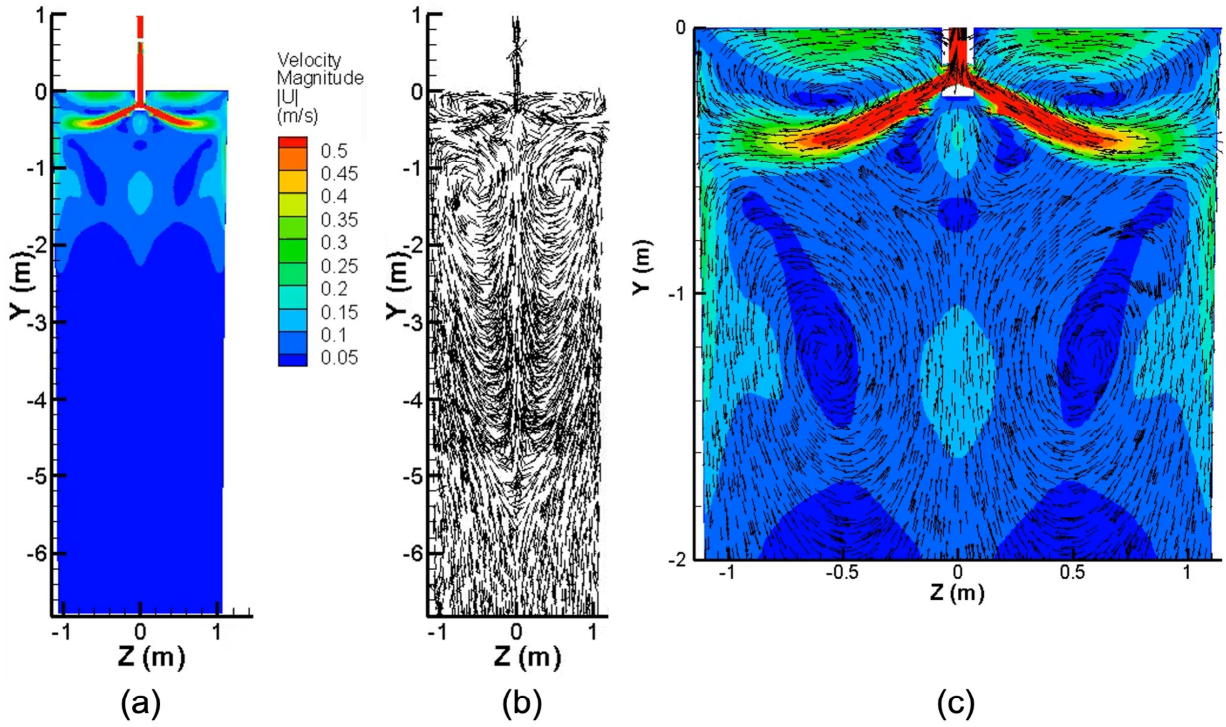


Figure 5.3 Initial steel flow pattern at t=0s with (a) velocity magnitude contour, (b) velocity vector, (c) close-up view of mold flow

Figure 5.4 and Figure 5.5 show the flow pattern at every 10s interval. At t = 0s in Figure 5.4 (a), the flow streamline is smooth because the solution is Reynolds-Averaged. After the LES-DPM coupled simulation starts, the streamline becomes curly due to formation of large eddies. at 20.56s, heat transfer is enabled as shown in Figure 5.4 (c), and at 30s, the solution starts becoming asymmetrical, with the right lower recirculation zone pushing to the left, as shown in Figure 5.4 (d). From 40s - 70s in Figure 5.5, small swirls in lower strand region occur, and the lower recirculation zone becomes shorter because thermal buoyancy dominates over inertia.

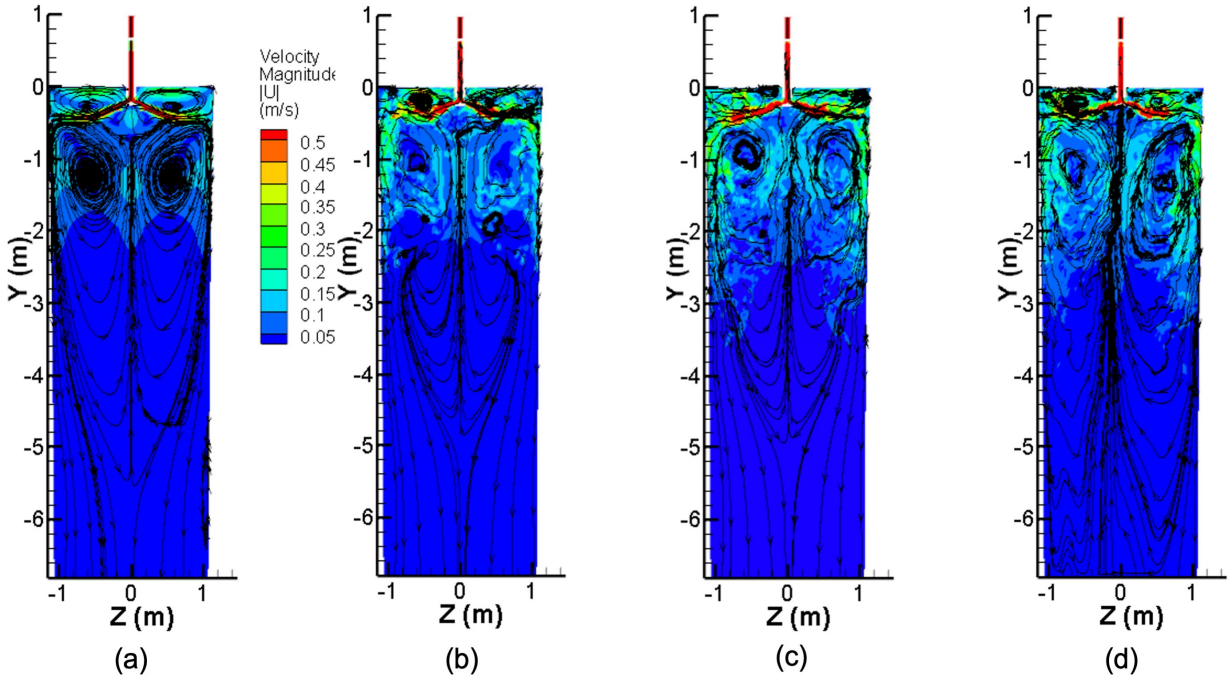


Figure 5.4 Flow pattern on center plane at (a) t=0s, (b) t=10s, (c) t=20.56s, (d) t=30s

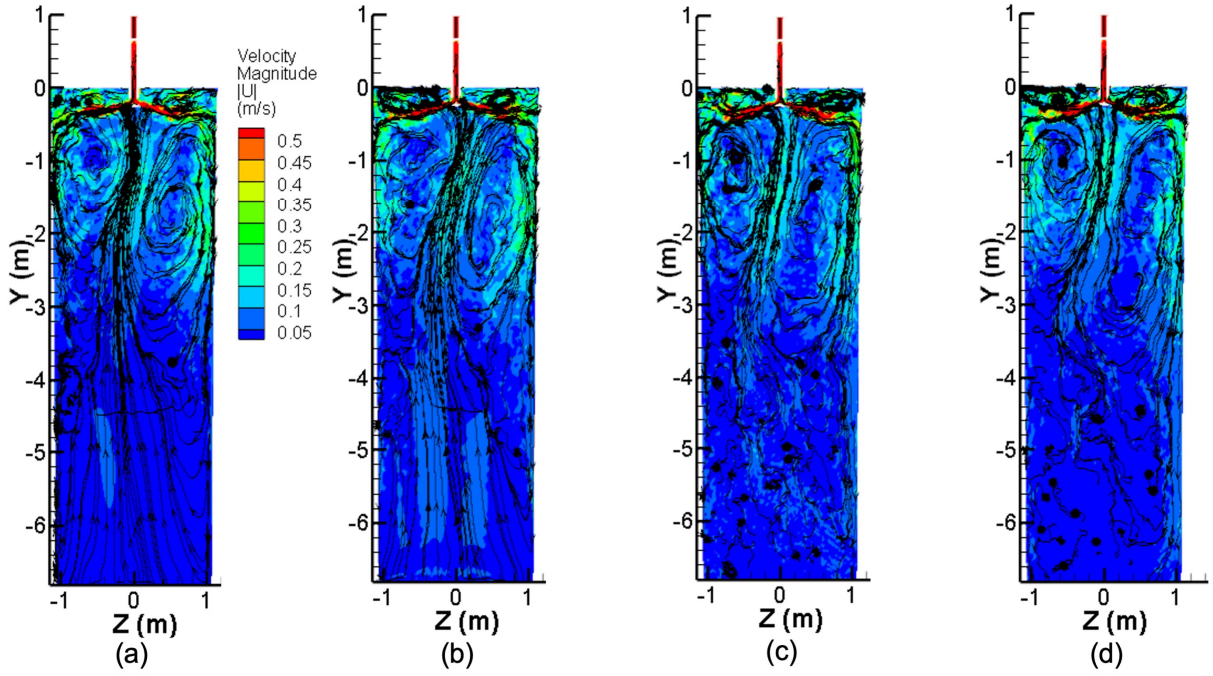


Figure 5.5 Flow pattern on center plane at (a) t=40s, (b) t=50s, (c) t=60s, (d) t=70s

Figure 5.6 compares the 37.6-second-averaged LES result with the RANS result. Apparently that the time-averaged LES flow is still asymmetrical and has back flow from the outlet. This means that the LES flow has not reached a quasi-steady state, even though it has been run for 6 months. More time is needed for this simulation in the future.

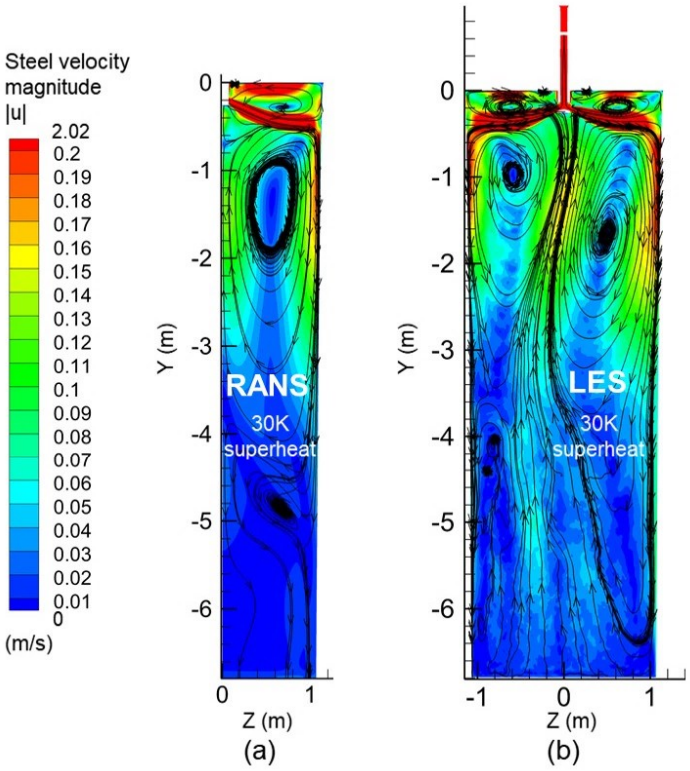


Figure 5.6 Time-averaged steel-velocity contour and streamline on center plane with (a) RANS and (b) LES model

5.5 Nail-board Validation

As shown in Chapter 4, the time-averaged LES results has not reached quasi-steady state, and needs more time. As discussed in Chapter 4, the average residence time in the strand, t_r, is 410 s, it would take $\sim$ 3 years with current computational power to have a fully-developed LES flow, especially deep down the caster, which is needed for accurate particle simulations. Furthermore, the second-order LES results does not match as well as the first-order RANS model, due to the extra diffusion in the real steel plant than numerical simulation.

5.6 Conclusions

This chapter uses a LES model to simulate the transient highly-turbulent and asymmetrical fluid flow in the nozzle and strand, initialized from a single-phase RANS result and enabling heat transfer at 20.56 s of simulation time. The results show that, at 30 s, the solution starts becoming asymmetrical, with the right lower recirculation zone pushing to the left. From 40 s - 70 s, small swirls in the lower strand region occur, and the lower recirculation zone becomes shorter because thermal buoyancy effect dominates over inertia effect. The time-averaged LES flow is still asymmetrical, and has back flow from the outlet. This means that the LES flow has not reached a quasi-steady state, even though it has been run for 6 months. More time is needed for this simulation in the future.

CHAPTER 6

MODELING OF MOLD-ELECTROMAGNETIC STIRRING IN A NOZZLE AND MOLD

6.1 Introduction

In continuous casting, argon gas is often injected through the nozzle refractory to prevent clogging and maybe entrapped into steel solidification shell, forming surface and internal defects. In this process, electromagnetic stirring in mold (M-EMS) can be used to rotate the flow around the mold perimeter to lessen particle entrapment, especially near the meniscus. The new three-dimensional multiphase computational fluid dynamics (CFD) model in Chapter 4 is extended to investigate the effect of M-EMS on flow pattern and particle transport and capture, by using a periodic boundary condition to simplify the geometry and modeling. The results show that the M-EMS pushes the nozzle jet downwards, leading to ∼30% more capture of 0.3mm-diameter argon bubbles (53%) than that without EMS (38%). However, M-EMS reduces the near meniscus bubble capture by the hook capture mechanism by ∼3 times due to higher velocity washing bubbles off.

6.2 Computational Models and Solution Procedure

The model system used here is similar to that described in detail in Chapter 4, with the addition of incorporating electromagnetic forces and an iso-thermal condition.

To start with, an iso-thermal steady-state Eulerian-Eulerian simulation with standard k-ϵ turbulence model is conducted for molten-steel-argon-gas multiphase flow. However instead of using a mean bubble diameter predicted by PFSG and two-stage bubble formation model as in Chapters 3 to 5, an empirical mean bubble diameter, 3 mm, is used in this study, because the previous method is only valid for bubbles horizontally injected into a vertical cross flow. In this study, argon gas is injected at the stopper rod tip through an inner conduit.

A time-averaged electromagnetic force filed is measured by the plant. This force field is interpolated onto the mesh as a User Defined Scalar (UDS), and the UDS in each cell is used as a source term in the Navier-Stokes equation for the steel phase as follows:

$$\alpha_s \frac{\partial \rho_s u_{is} u_j}{\partial x_j} = -\alpha_s \frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}[(\mu_s + \mu_{ts})(\frac{\partial u_{is}}{\partial x_j} + \frac{\partial u_{js}}{\partial x_i})] - R_{igs} + \sum F_{is} + F_{iEMS} + S_{is-momentum-sink} \tag{6.1}$$

Then, 10,000 3 mm-diameter argon bubbles are injected and tracked with a Discrete Phase Model (DPM) and a Random Walk Model (RWM) into this steady-state Eulerian-Eulerian velocity and

turbulence field in a one-way coupled way. The capture location and fraction are determined by the hook / entrapment / engulfment capture mechanism. The shell thickness and PDAS size are the same as those in Chapter 4. The hook depth d_{hook} and critical capture time t_c is set as 6 mm and 0.44 s.

6.3 Flow Model Domain, Mesh, and Boundary Conditions

The flow model domain, including the complete stopper rod, nozzle, and liquid pool in the strand, is shown in Figure 6.1 (a). Only half of the domain is modeled due to its symmetry about the center plane. A 0.5-m-tall and 0.5-m-diameter cylinder is built around the stopper rod to simulate the flow radially coming in from the tundish well chamber. The vertical strand is 3.55-m-long, and the liquid pool width and thickness down the meniscus is built based on the solidifying steel shell evolution shown in Chapter 4.

The mesh is created in ANSYS Workbench and includes ~0.75 million structured hexahedral cells for nozzle and strand liquid-pool domain altogether, as shown in Figure 6.1. The cell volume ranges from $8.9 \times 10^{-}9$ to $5.6 \times 10^{-}6m^3$, which corresponds to a cell length from 2 to 18mm. The Y^+ value varies from 2.9 to 296 in the strand, so the standard wall-function is used. Figure 6.1 (b) shows that the mesh going through the gap between stopper rod and SEN is refined due to large velocity gradient. The mesh in the strand is aligned with the nozzle-jet direction by having a downward angle after the nozzle port and then being flat, in order to minimize false dissipation, as shown in Figure 6.1 (c).

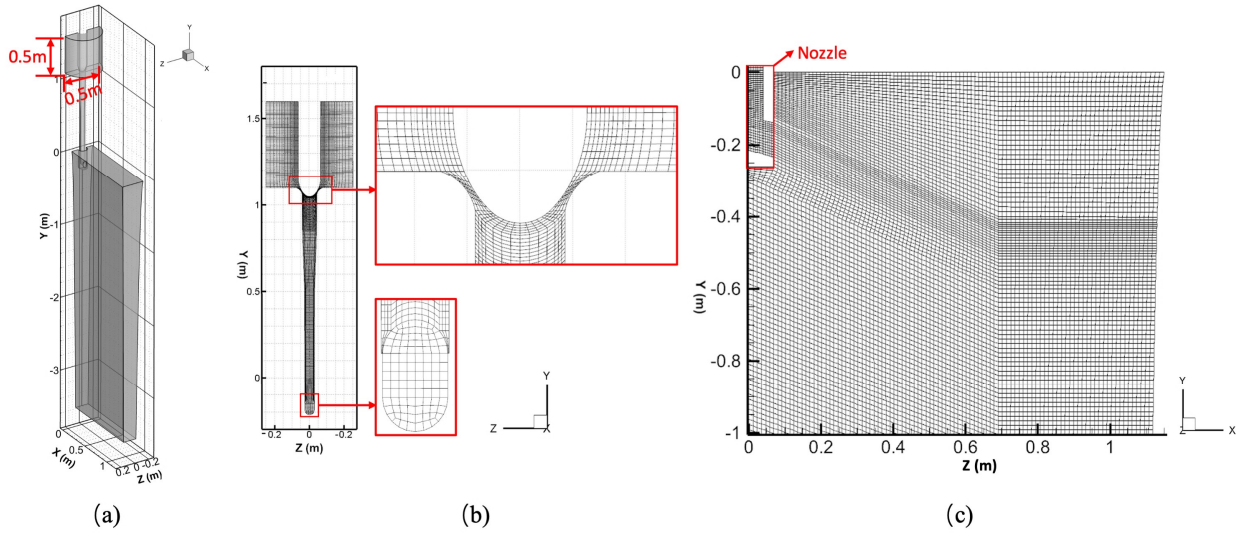


Figure 6.1 Geometry and mesh for (a) whole domain (b) nozzle center plane and (c) strand center plane

Boundary conditions with EMS off (flow-model step 1), EMS on (flow-model step 2) and bubble transport and capture (flow-model step 3) simulations are given in Figure 6.2. Steel flow is coming into the upper cylinder radially at a fixed velocity of 0.0087 m/s, calculated from the steel volume flow rate divided

by the inlet area. The turbulence kinetic energy k, its dissipation rate ϵ are fixed at $10^{-5}m^2/s^2$ and $10^{-5}m^2/s^3$ at the inlet. The argon gas volume fraction and an empirical mean bubble size, 3mm, are chosen for argon gas injection, through the tip of stopper rod, at the locations shown in Figure 6.2 (a) and (b). The argon gas is injected at 8.275 m/s, based on the hot argon volume flow rate divided by the inlet area. Nozzle wall has a 0.001m roughness and has non-slip boundary condition. The top surface is approximated as no-slip for the molten steel flow, due to the large viscosity of the slag layer. This surface allows argon bubbles to escape (due to the upward flotation from their buoyancy as they flow with the molten steel). Other surfaces (the wide and narrow faces) are also set as non-slip walls with mass, momentum, and energy sink UDFs on their adjacent cells. A pressure outlet condition is chosen over the strand domain bottom, according to the ferrostatic pressure head of molten steel ($\rho_s gh$), where h is the distance below the meniscus. The outlet pressure is calculated by a UDF in Appendix A. The turbulent kinetic energy and its dissipation rate are set as $k = 10^{-5}m^2/s^2$ and $\epsilon = 10^{-5}m^2/s^3$ for reversed flow. For the situation with EMS off in Figure 6.2 (a), the center X = 0 m plane is treated as symmetry plane where the gradients of all variables are 0. This simplifies the geometry and flow by enabling simulating only half of the domain. With EMS on, because the EMS force is 180 degree-rotation symmetric about the coordinate original point, (X,Y,Z) = (0 m,0 m,0 m), the center plane (X = 0 m) in the strand part is set as a pair of periodic boundaries where the flow variables flowing out of one periodic plane is equal to the variables coming into the other periodic plane. The nozzle-part center plane is still set as a symmetry plane because the EMS force is assumed to have no effect on the nozzle flow due to its high downward velocity and weak electromagnetic (EM) force in the mold center region.

Finally, the DPM model is run with 10,000 3mm-diameter argon bubbles. The injected bubbles can either escape from the top surface or be captured by the solidification shell (entrapment / engulfment), the hook capture mechanism, or later deep in the strand, if they exit the outlet or still stay in the domain at the end of simulation, as shown in Figure 6.2 (c). The time step size is 0.005s to meet the Courant number criterion.

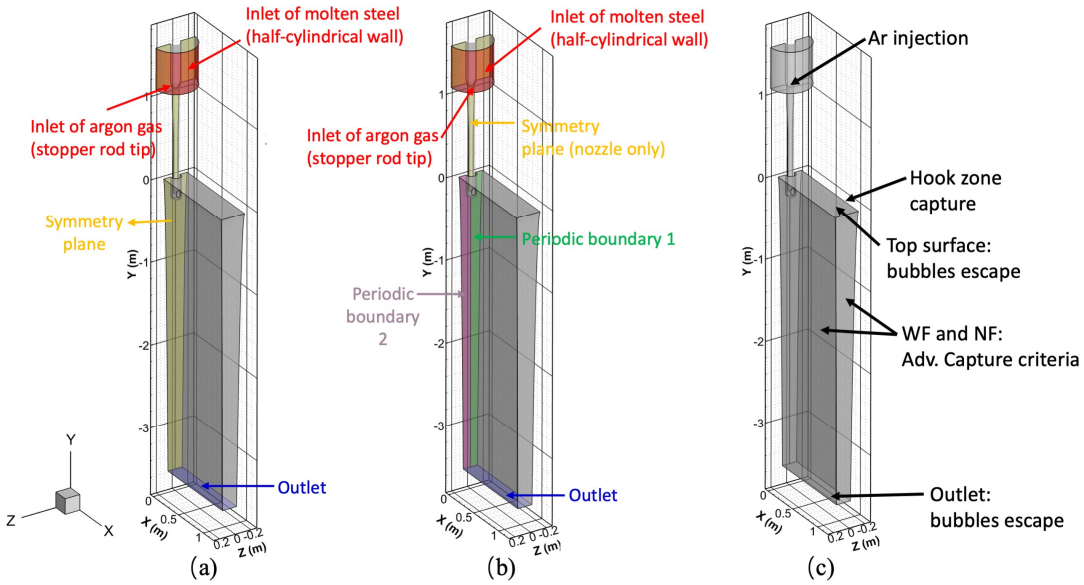


Figure 6.2 Boundary conditions for (a) step1: EMS off, (b) step2: EMS on, (c) step3: bubble transport and capture simulations

Because the second-order upwind scheme caused divergence of the solution, first-order upwind scheme is used throughout every step. The coupled scheme is used to solve the pressure-velocity coupling equations, which solves all equations for phase velocity corrections and shared pressure corrections simultaneously and has been proved to be efficient in steady-state simulation [114]. The simulations of first two steps are conducted on a high-performance computer Mio with 48 computational nodes. Convergence is obtained when all residuals reached 10^{-4}. The computational time is $\sim$ 10 hours for each step. A Dell computer tower with 10 computational nodes was used for step 3, and it takes $\sim$1 day for less than 0.05% of bubbles to be still in the domain.

6.4 Electromagnetic Force Distribution

A time-averaged three-dimensional EM force distribution per cubic meter on a 357mm x 2300mm x 5m ($t \times w \times l$) straight slab is provided by the plant in the form of:

Table 6.1 Electromagnetic Force Distribution

X (m)	Y(m)	Z(m)	$F_x(N/m^3)$	$F_y(N/m^3)$	$F_z(N/m^3)$
⋮	⋮	⋮	⋮	⋮	⋮

Because the domain in this study is 300-mm-thick, the Y coordinate in Table 6.1 is divided by 1.19 to fit the electromagnetic force data onto the domain. The adjusted data is then interpolated onto the mesh

and stored as a User Defined Scalar (UDS) in each cell as shown in Figure 6.3. For Fx, its direction is from nozzle to narrow face on the +Z side and is opposite on the -Z side. Therefore, the EM force tends to stir the steel flow counterclockwise. The UDS is then used as a source term, F_{iEMS}, for the steel phase.

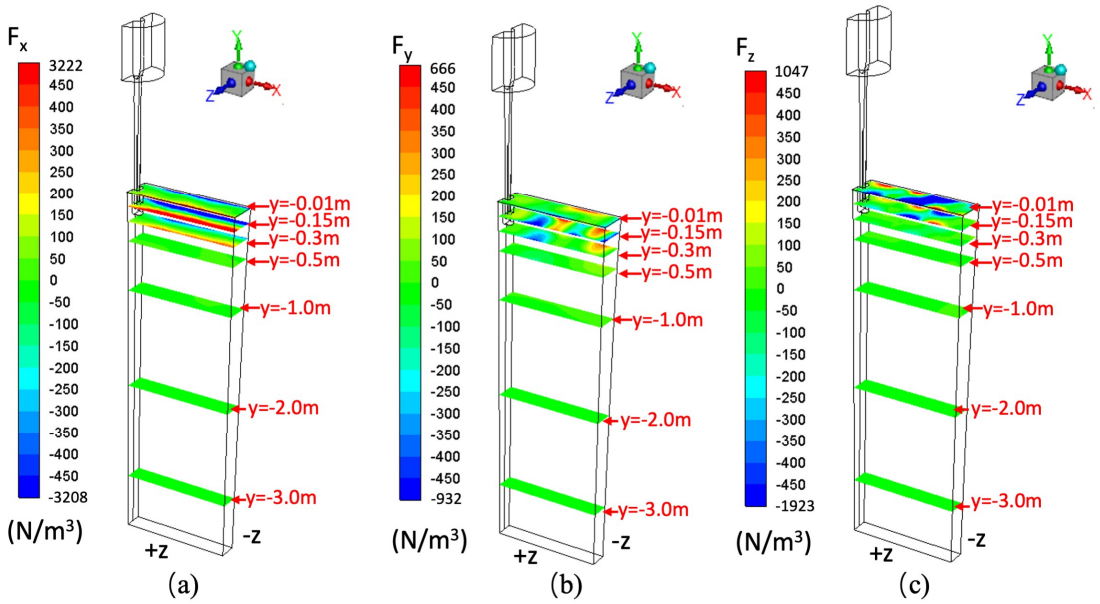


Figure 6.3 Electromagnetic force distribution per unit volume on different planes below meniscus for (a)F_x, (b)F_y, and (c)F_z

6.5 Casting Conditions

Two conditions, with and without M-EMS, are simulated and compared with the caster dimensions and casting conditions in Table 6.2.

Table 6.2 Caster Dimensions and Process Conditions for Half of a Caster

Parameters	Symbols	
Port down angle	θ	15°
Slab thickness/width (mm)	t_s/w_s	300/2300
Tundish height (mm)	h_t	1400
Casting speed (m/min)	v_c	0.6
Steel flow rate (metric ton/min)	m	1.45
Submergence depth (mm)	d_s	140
Argon flow rate: Hot/Cold (LPM/SLPM)	Q_h/Q_s	19.2/8
Hot argon gas density (kg/m^3)	ρ_h	0.67
Argon volume fraction	f_{Ar}	4.4%
Molten steel density (kg/m^3)	ρ_s	7090
Molten steel dynamic viscosity (kg/m·s)	μ_s	0.0063

In this study, the effect of M-EMS on flow pattern and particle capture has been numerically investigated on a stopper-rod-controlled caster. The stopper rod is lifted vertically by 11.67mm from its fully closed position by linearly fitting plant measurement data (steel flow rate vs. stopper rod opening).

6.6 Hot Argon Volume Flow Rate

Because the argon gas is injected from room temperature and pressure into molten steel with a temperature $\sim 1826K$ and a hydrostatic pressure, its volume expands in this process. the expansion factor, E_f can be calculated from the ideal gas law and is defined by:

$$E_f = \frac{T_h \times P_s}{T_s \times P_h} \tag{6.2}$$

where T_s is the standard temperature (273K), T_h is the casting temperature (1826K), P_s is the standard pressure (101325Pa), and P_h is the hot steel-argon shared pressure and can be predicted by FLUENT. The static pressure distribution with EMS off is shown in Figure 6.4. The lowest pressure occurs at the gap between stopper rod and nozzle due to sudden contraction. There's a low-pressure zone right below the stopper rod tip, indicating a possible gas pocket there, and bubbles will be sheared off the big gas pocket by the high-velocity flow around it, forming a bubble size distribution. The static pressure increases with the increase depth below stopper rod and meniscus due to increasing ferrostatic pressure. Then P_h along four lines (vertical lines 1–2, 2–3, 4–5, and 5–6 across the nozzle and strand) in Figure 6.5 are extracted from Figure 6.4 to calculate the expansion factor, E_f, as shown in Figure 6.5.

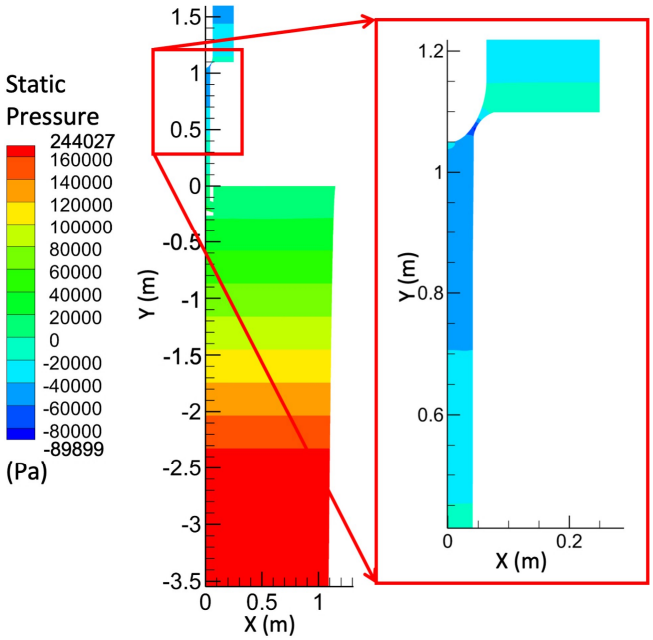


Figure 6.4 Pressure distribution on domain center plane with EMS off

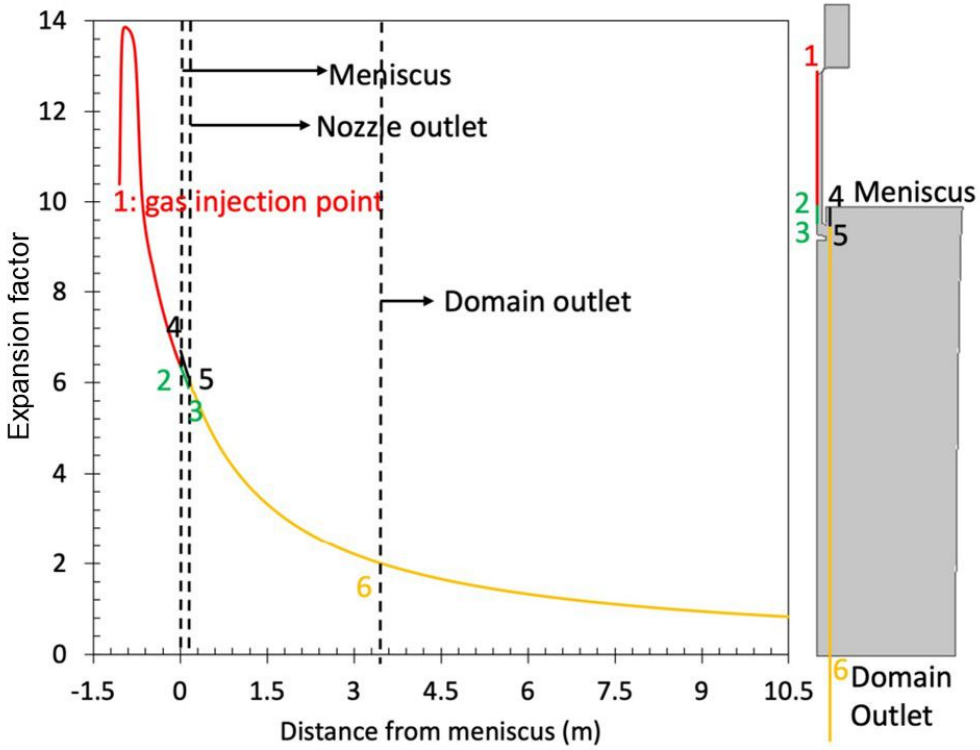


Figure 6.5 Distribution of expansion factor with respect to distance below meniscus

The E_f goes all the way up to ∼14 near the injection point, i.e. stopper rod tip, due to the low pressure caused by the gas pocket. at the domain outlet, E_f goes down to ∼ 2 due to the large ferrostatic pressure. In this study, E_f is set at 2.4 to minimize argon gas buoyancy effect on the mold flow, which leads to most bubble capture as bubbles tend to go deeper.

The argon volume fraction is calculated with Q_h: $Vf_{Ar} = \frac{Q_h}{Q_h+Q_{steel}} = 4.4\%$, where Q_{steel} is the steel volume flow rate in m^3/s.

6.7 Flow Results

Figure 6.6 shows molten steel-argon gas flow patterns with colors of flow velocity magnitude in the whole domain and nozzle part. Both situations show a classic double-roll flow pattern. The nozzle flow patterns appear to be the same with and without EM force due to its high velocity. In the mold part, the jet in pushed downwards with EMS on, and the impingement point from the jet on narrow face goes deeper, which is in line with the findings from X. Sun et al.[119]. This downwards flow means that more bubbles will be transported and captured deeper down the strand.

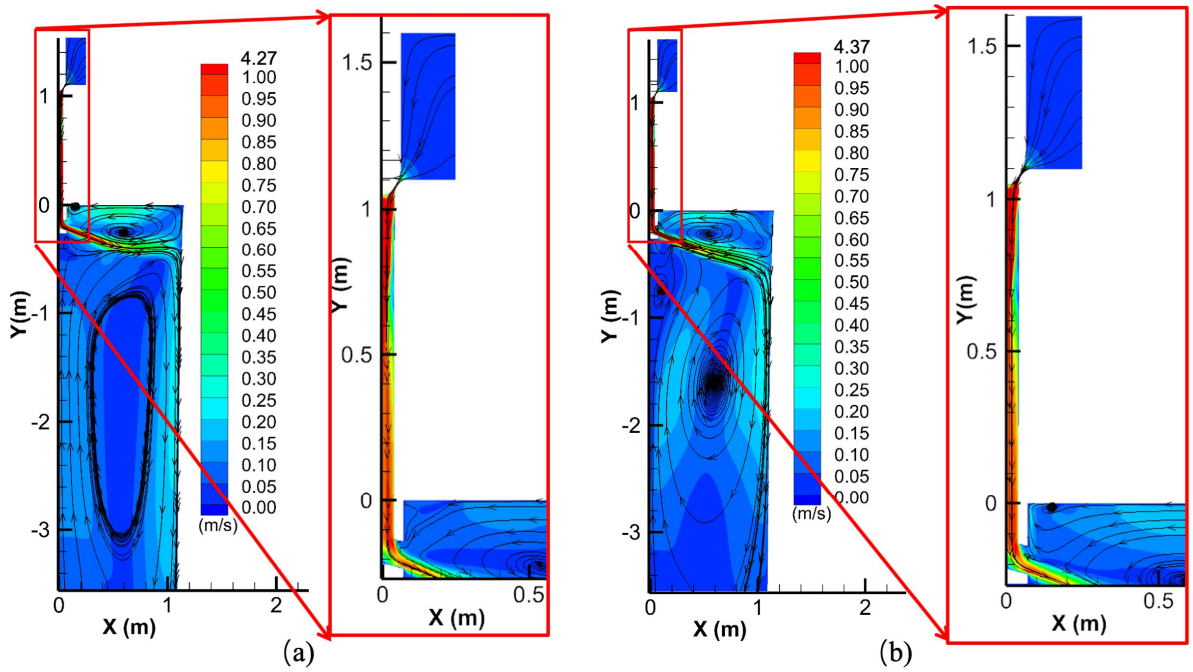


Figure 6.6 Flow pattern in the whole domain and close-up view of nozzle with (a) EMS off and (b) EMS on

Figure 6.7 shows a close-up view of flow near the stopper rod. The velocity goes above 4m/s for both situations due to the sudden contraction of area that molten steel can go through.

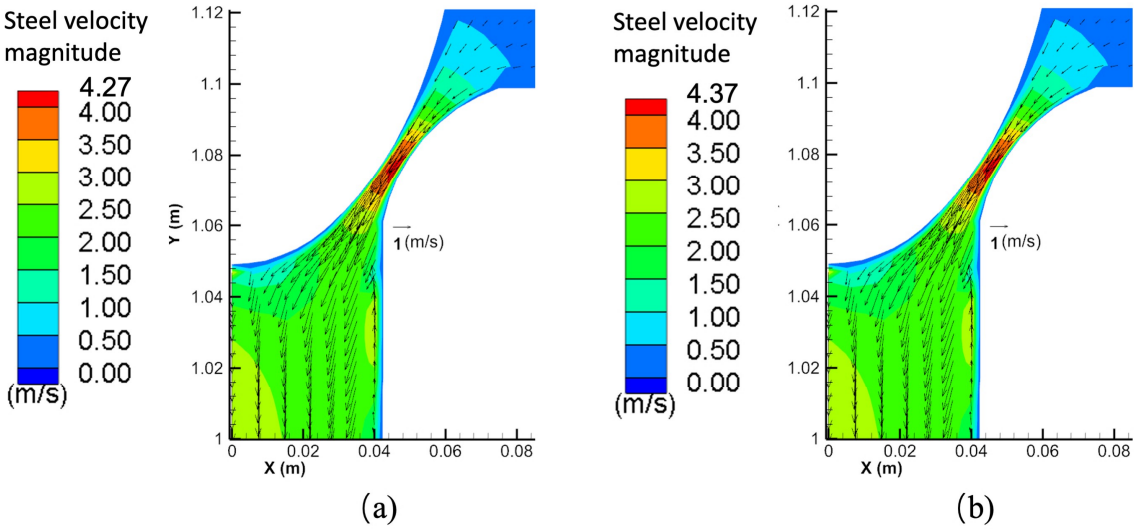


Figure 6.7 Flow pattern near the stopper rod with (a) EMS off and (b) EMS on

Figure 6.8 shows the steel x velocity and velocity vectors on the nozzle port with and without EM force. The EM force has negligible effect on flow patterns of the jet zone, i.e. the red high-velocity region near the

lower half of the nozzle port. However, the EM force makes the back flow zone move towards -Z direction because flow inertia is not dominant over there.

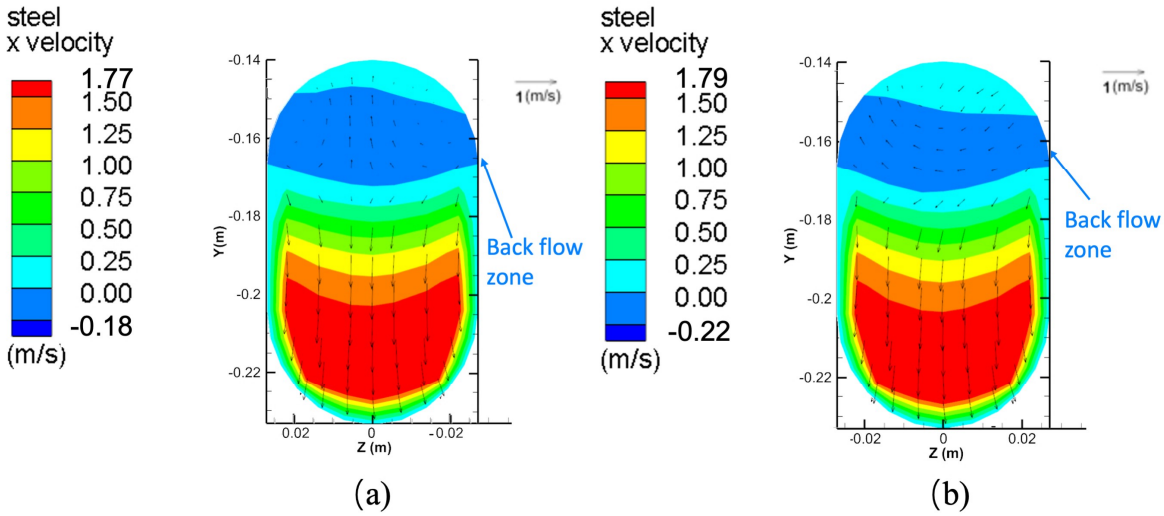


Figure 6.8 Steel x velocity contour and velocity vectors on nozzle port with (a) EMS off and (b) EMS on

The jet characteristics, including jet velocity and angle, and turbulence, are calculated at the nozzle port outlet planes for both cases and are given in Table 6.3. The jet is defined only where there is positive outflow from the nozzle, and details are given elsewhere [71]. The jet characteristics are not affected by the M-EMS.

Table 6.3 Jet characteristics at nozzle ports

Parameters	Symbols	EMS off	EMS on
Weighted average liquid velocity at the nozzle port in x-direction	$\bar{U}_{xs}$	1.52m/s	1.52m/s
Weighted average liquid velocity at the nozzle port in y-direction	$\bar{U}_{ys}$	0.66m/s	0.66m/s
Weighted average liquid velocity at the nozzle port in z-direction	$\bar{U}_{zs}$	0.004m/s	0.0003m/s
Weighted average turbulence kinetic energy at the nozzle port	$\bar{K}$	$0.05\text{m}^2/\text{s}^2$	$0.05\text{m}^2/\text{s}^2$
Weighted average turbulence dissipation rate at the nozzle port	$\bar{\epsilon}$	$0.87\text{m}^2/\text{s}^3$	$0.87\text{m}^2/\text{s}^3$
Vertical jet angle	θ_{yz}	23.4°(downward)	23.5°(downward)
Horizontal jet angle	θ_{xz}	89.8°	90.0°
Jet speed	U_{jet}	1.66m/s	1.65m/s
Back-flow zone fraction	η	20.72%	19.63%

Figure 6.9shows the steel velocity and vectors on three planes, 0.01m, 0.075m and 0.15m below meniscus with and without EM force. Figure 6.9 (a) - (c) show that the flow is uniformly moving from NF

towards nozzle, and the other flow is moving from nozzle to NF due to argon gas flowing up. With EMS on, the flow patterns are significantly changed: the EM force accelerates the flow in the same direction on the -Z side and decelerates the flow in the opposite direction on the +Z side, as shown in Figure 6.9 (d) - (f).

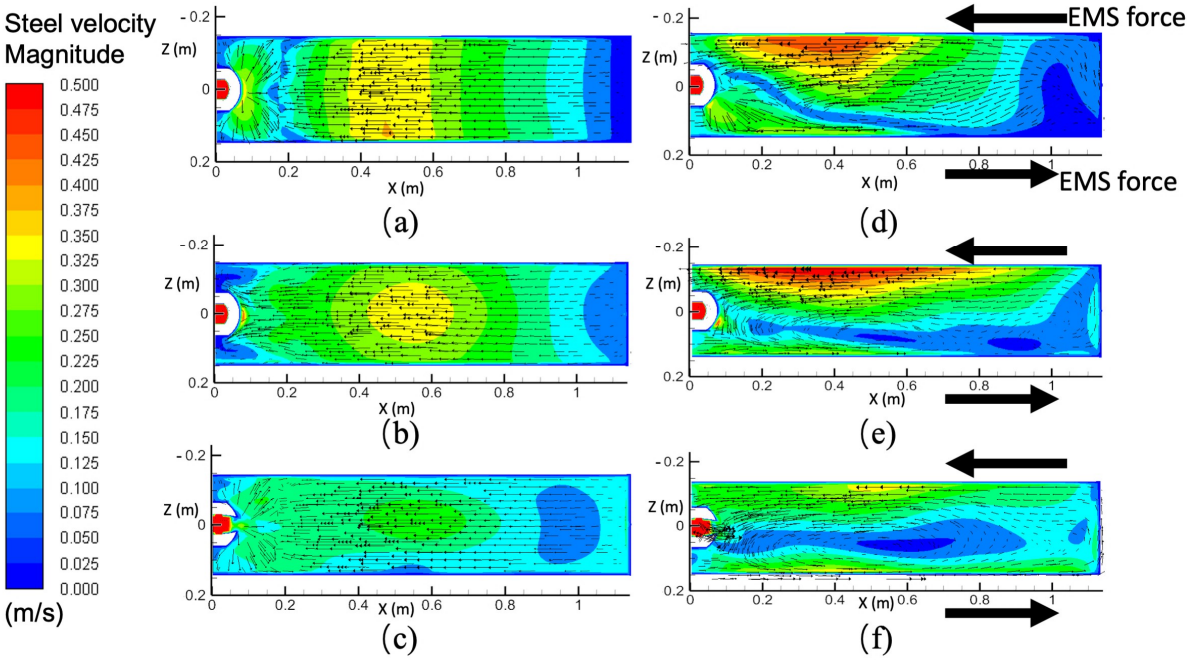


Figure 6.9 Steel velocity magnitude and velocity vector with EMS off on planes (a) Y=-0.01m, (b) Y=-0.075m, and (c)Y=-0.15m; with EMS on planes (d) Y=-0.01m, (e) Y=-0.075m, and (f)Y=-0.15m

6.8 Argon Volume Fraction Results

Figure 6.10 shows the argon volume fraction distribution on the center plane with and without EM force. Above the nozzle port and in the mold, EMS makes the argon volume faction lower because the jet is pushed downwards and tends to transport argon further towards the narrow face. The close-up views near the stopper rod show that both situations exhibit two gas pockets: one is under the stopper rod due to argon gas injection, and the other one is caused by the recirculation zone shown by the black arrows. The high downward steel flow will shear off small bubbles off these two gas pockets, forming a bubble size distribution.

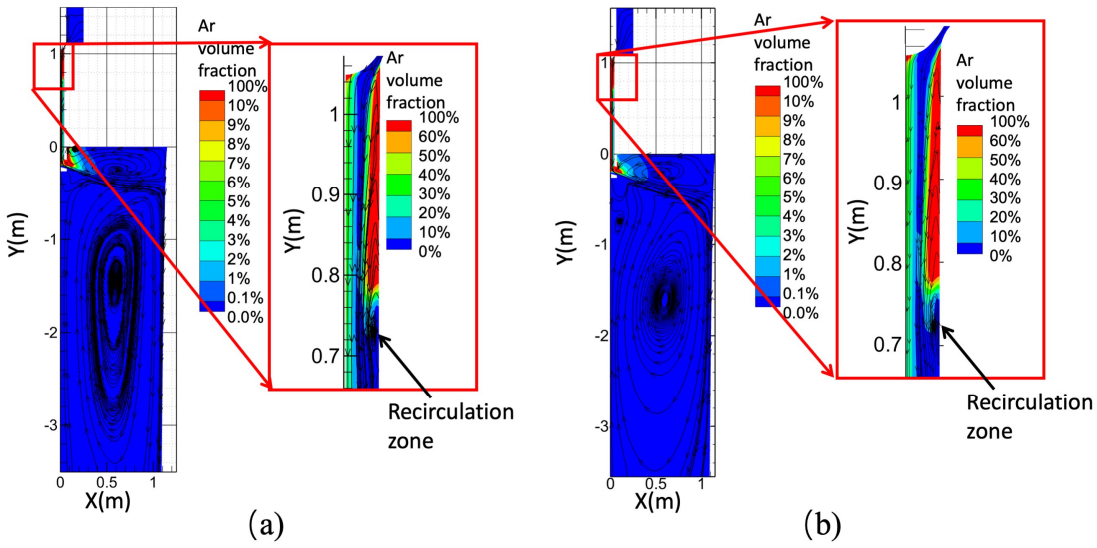


Figure 6.10 Argon volume fraction at center plane with (a) EMS off and (b) EMS on

6.9 Top Surface Level Results

An equation was developed by Q. Yuan[18] from potential energy balance to calculate the surface steel height with respect to its quiescent state where the height is defined as 0mm. The equation is as follows:

$$\triangle z(x,y) = \frac{p(x,y) - p_{mean}}{(\rho_s - \rho_{top})g} \times 1000 \tag{6.3}$$

where $\triangle z(x,y)$ is the top surface level in mm, $p(x,y)$ is the local surface pressure in Pa, p_{mean} is the top surface area-averaged pressure in Pa, ρ_s is steel density (7090kg/m^3), ρ_{top} is the slag density (2600kg/m^3), and g is the gravitational acceleration, 9.81m/s^2. The mean pressures with EMS off and on are calculated by FLUENT, 810 Pa and 774 Pa respectively.

Figure 6.11 (see page 97) shows the top surface level distribution. With both conditions, the top surface level is higher near narrow face and nozzle wall, because the steel hits the wall and gets deflected upwards. Surface level near mold center is lower to compensate for the higher level near the walls. The maximum surface levels are both near the nozzle wall, because of argon gas flowing up. The maximum surface level decreases by 66% from 24 mm to 8 mm with EMS on.

Figure 6.12 (see page 97) shows the surface level along the two centerlines in Figure 6.11 for situations with and without EM force. The surface level is up to 24mm near nozzle wall and 12 mm near NF with EMS off. EMS brings the numbers down to ~5 mm, meaning possible less slag entrapment near the top surface.

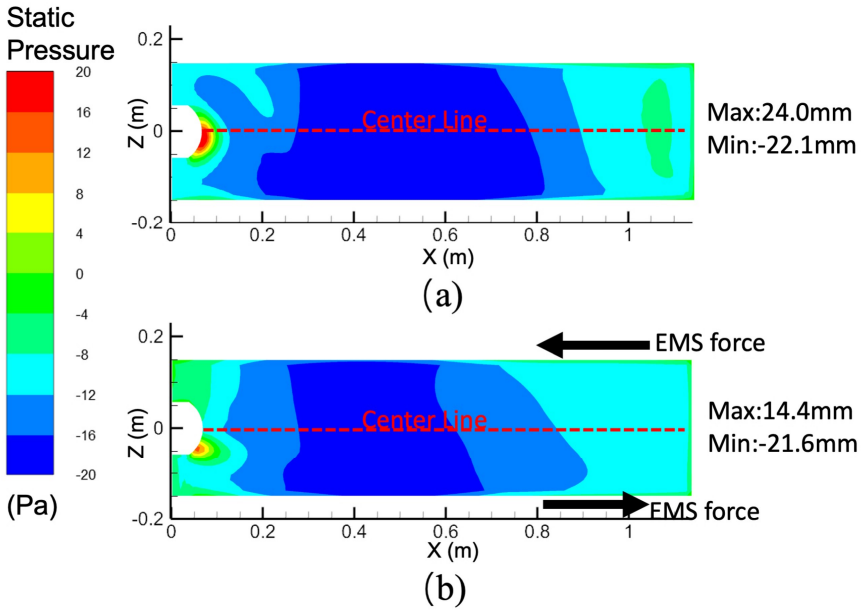


Figure 6.11 Top surface level distribution with (a) EMS off and (b) EMS on

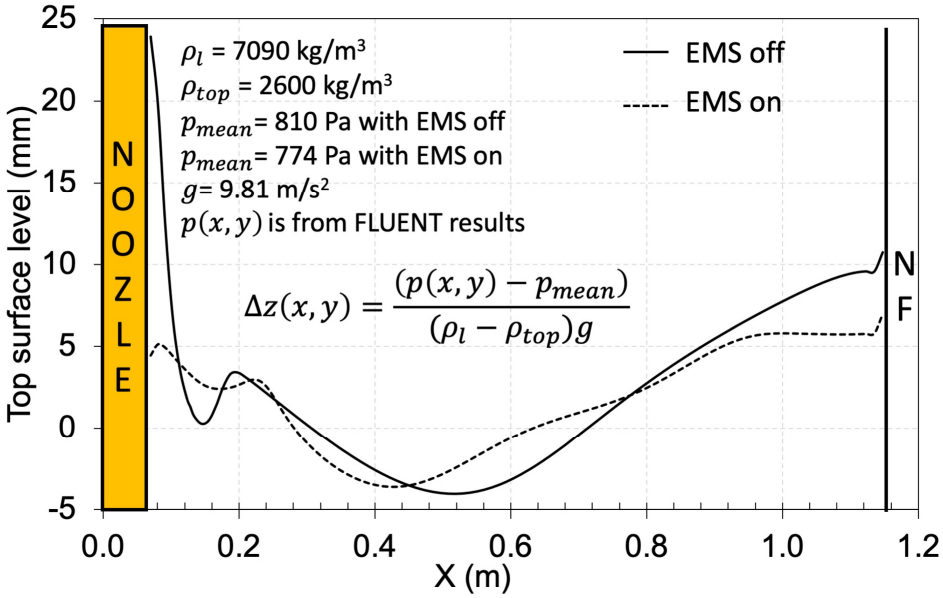


Figure 6.12 Top surface level distribution along center lines

6.10 Particle Transport and Capture Results

Figure 6.13 shows the locations where the 10,000 3mm-diameter argon bubbles are captured on wide and narrow faces with and without EM force. With EMS on, more bubbles are captured deeper down the strand, because the EMS force pushes the jet down.

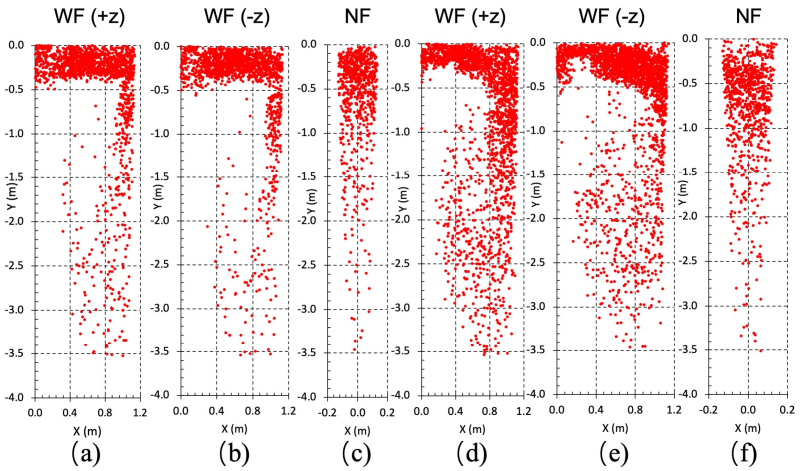


Figure 6.13 Bubble capture locations with EMS on (a) WF(+z) (b) WF(-z) (c) NF; with EMS off (d) WF(+z) (e) WF(-z) (f) NF

Finally, Figure 6.14 shows the breakdown of bubble capture fraction at different locations of the strand with and without EM force. EMS increases the bubble capture fraction on both WF and NF due to the jet being pushed down, which transports more bubbles deeper into the strand. This means that less bubbles escape into the top slag layer harmlessly with EMS on, and the fraction of bubbles escaping into the top surface decreases from 59% to 46%.

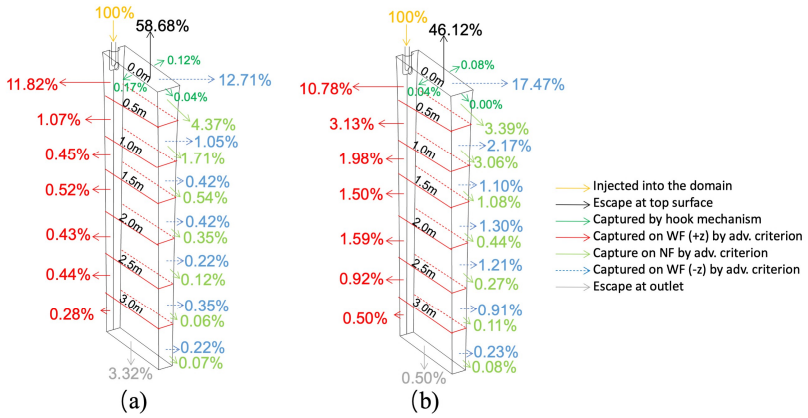


Figure 6.14 Bubble capture fraction breakdown, every 0.5m below meniscus with (a) EMS off and (b) EMS on

The capture fraction by entrapment / engulfment criterion in the first 0.5mm below meniscus is higher on WF(-z) (0.08%) than WF(+z) (0.04%) with EMS on, possibly because the higher speed towards nozzle on the -z side tends to balance the x-direction momentum of argon bubbles moving towards NF, and then bubbles are captured due to force balance. Even though the capture fraction on WF and NF is higher with

EMS on, the capture fraction by hook mechanism is significantly reduced from 0.33% to 0.12%, which is ∼3 times lower. This is because the high steel velocity tends to wash the bubbles away from the hook zone, so they cannot stay long enough inside and be captured.

Figure 6.15 shows the capture percentage of bubbles by entrapment / engulfment criterion on WF(+Z), WF(-Z), and NF. On each face, capture percentage is higher with EMS on due to jet being pushed down. Overall, ∼30% more 0.3mm bubbles are captured with EMS on (53%) than with EMS off (38%) by all three mechanisms.

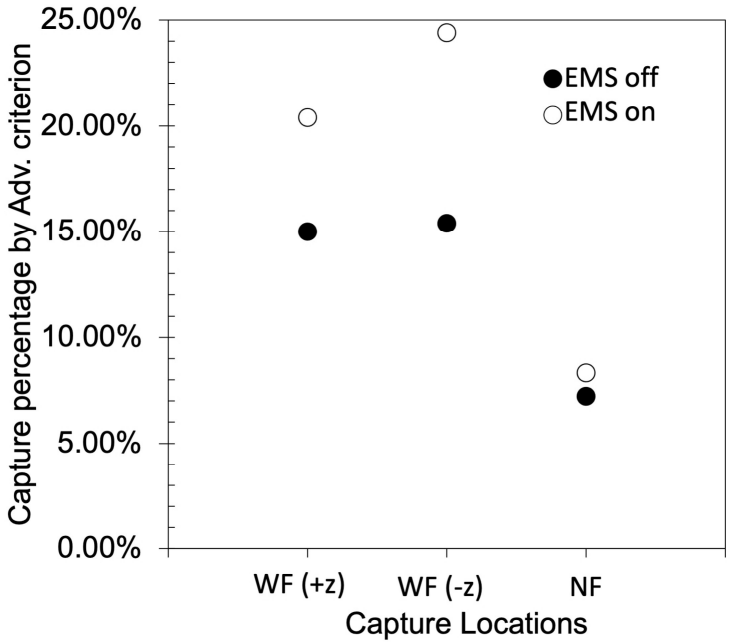


Figure 6.15 Bubble capture percentage by entrapment / engulfment criterion on WF(+z), WF(-z) and NF with and without EM force

The hook depth in this study is set as 6mm. Figure 6.16 shows the bubble capture percentage by both entrapment / engulfment criterion and hook mechanism within 6mm below the strand surface. The capture percentages within 6mm below the strand surface (hook zone) by both capture mechanisms are lower with EMS on, caused by the high steel velocity near the meniscus washing bubbles away from the dendritic solidification front and hook zone. With EMS on, the capture percentage by entrapment / engulfment criterion is reduced from 1.66% to 1.14% and is 3× smaller by hook mechanism.

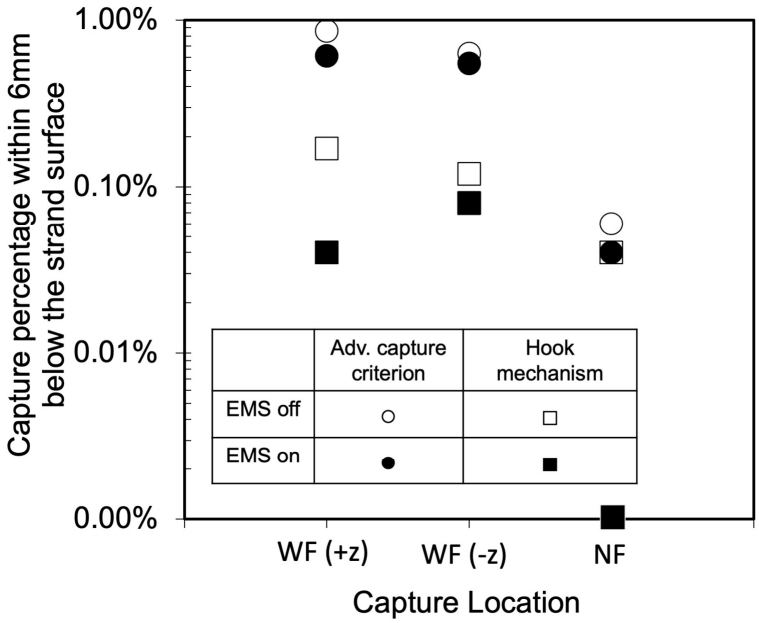


Figure 6.16 Bubble capture percentage within the hook zone by both entrapment / engulfment criterion and hook mechanism on WF(+z), WF(-z) and NF, with and without EM force

6.11 Conclusion

The new modeling approach in Chapter 4 has been extended to investigate the effect of M-EMS on multiphase turbulent flow, argon gas injection, and particle transport and capture in the strand during steady continuous casting of steel slabs, with a periodic boundary condition. 10,000 3mm-diameter argon bubbles care injected and tracked into this steady-state flow pattern.

The results suggest:

1. Effect of EMS force on flow pattern in nozzle is negligible, due to the small EM force in mold center region and high velocity in nozzle which dominates the flow.

2.EMS causes impingement point on NF to go deeper, because the EM force pushes the flow down.

3. The counterclockwise EM force causes molten steel mold surface flow to accelerate where EM force adds to flow from NF to nozzle and to decelerate where EM force cancels flow from nozzle to NF.

4. With EMS on, fewer bubbles are captured in hook zone (6mm below the strand surface) by advanced capture criterion and hook mechanism respectively, due to high velocity near WFs. But $\sim 30\%$ more bubbles are captured by all mechanisms with EMS (53%) than without EMS (38%), because EM force pushes the jet down.

One way to improve this model is to use the more realistic EEDPM model developed by H. Yang[117] which combines the Eulerian-Eulerian multiphase model with the Discrete Phase Model (DPM) in a two-way coupled way. The bubble expansion, coalescence, and breakup are considered in the DPM model, which can be used to determine the local bubble size in the continuous phase of Eulerian-Eulerian model at each time step. A more realistic flow pattern and bubble size distribution can be obtained this way. Other than improving the computational model, more plant measurements are also of great helpful for validation.

CHAPTER 7

EFFECT OF LADLE STREAM SUPERHEAT ON TUNDISH FLOW PATTERN AND INCLUSION TRANSPORT AND REMOVAL

7.1 Abstract

As shown in Chapter 4, higher superheat leads to less and shallower capture of argon bubbles due to creation of multiple recirculation zones which lessens penetration. However, this effect can be balanced out or gets worse if the successive ladles have lower superheat, which changes the flow pattern in the tundish and naturally increases the number of inclusions exiting the tundish outlet. In this chapter, a three-dimensional non-isothermal computational fluid dynamics (CFD) model with Eulerian-Lagrangian one-way-coupled method is developed, to investigate the effect of pouring 10 K-superheat colder steel into 30 K-superheat hotter steel on the internal fluid flow in a furnished two-strand-caster tundish and inclusion removal. Results show that the cold steel turns the three top-surface-directed upwards jets to flow along the bottom of the tundish, which short-cuts the flow path to the tundish outlet and deteriorates the removal of inclusions. It takes 2,700s for the flow and temperature to reach quasi-steady state again, which makes the steel quality worse on a 19-m-long slab. Smaller the inclusion, higher the exit fraction at tundish outlet; feeding 20K colder steel into hotter steel increases the exit fraction for small inclusions ($\leq$100μm) and large inclusions (>100μm) by 2-3 times and 32-80 times, respectively.

7.2 Introduction

Tundish, served as a buffer zone and regulator of steel flow between the ladle and continuous casting molds, plays a vital role in steel cleanliness by removing the inclusions from upstream ladle reoxidation process into its top slag layer. During this process, the hotter or colder molten steel will be poured from ladle into tundish. The temperature of the inlet stream from the ladle may vary from different heats, or on the ladle's teeming time [52]. The thermal buoyancy effect induced by temperature difference between liquid steel may change the flow pattern inside tundish significantly and affects the flotation and removal of inclusions.

Therefore, there is a great incentive to study this transient behavior during ladle exchange, especially from exchange from hot steam to cold stream because the colder molten steel tends to move downwards when feeding into hotter steel due to its higher density, thus changing the flow pattern and inclusion removal drastically.

Extensive research has been done to investigate the non-isothermal fluid flow in tundish with numerical modeling and water experiments, but with a focus mainly on optimization of flow control device and its effect on inclusion removal [50, 53–59]. Joo et al. (1993) [53] proposed that even a small temperature drop ($1°C$) would make natural convection currents significant in a 1.1-m-deep tundish, particularly at the end of tundish walls, owing to small flow velocities and large heat losses in these regions. They also concluded that small inclusions ($\leq 40\mu$m) are less readily removed into slag layer due to its small stoke rising velocity and tendency to move with the flow.

Far few studies are conducted on the effect of inlet-stream temperature rise or drop on the effect of tundish flow pattern and inclusion removal. Chakraborty and Sahai (1991) [60] developed a transient, three-dimensional model to study the effect of varying ladle temperature on the fluid flow and heat transfer phenomena in a typical twin strand slab caster tundish. With an incoming steel temperature declining at a rate of $0.5°C/min$ over a casting period of 50min, the cooler incoming steel starts to flow along the bottom of the tundish instead of the normal top-free-surface-directed flow. Thus, the inclusion flotation and removal are expected to be worse. Q. Tian et al. (2012) [61] investigated the effect of both $0.5°C/min$ and $0.25°C/min$ inlet cooling rate on the fluid flow and heat transfer in a flexible thin slab casting tundish. With $0.5°C/min$ cooling rate, the horizontal flow along the top surface turns to flow along the bottom of tundish, which shortens the fluid flow routes and deteriorates the removal of non-metallic inclusions from molten steel; With $0.25°C/min$ cooling rate, the horizontal flow sustains through the whole casting process. However, both studies did not quantify the effect of fluid flow on inclusion removal.

Chatterjee and Chattopadhyay [62] used both water experiment and numerical modeling to study the effect of non-isothermal conditions on fluid flow in a multi-strand billet caster tundish. It was observed that using inlet temperature step down by $10°C$ or up by $23°C$ for water and steel, respectively, changed the fluid patterns significantly, especially in regions far away from the inlet due to less inertia effect[62]. The incoming hotter fluid in case of step-up condition resulted in an upward buoyant flow which aided flotation of inclusions, whereas the incoming colder fluid in case of step-down condition dragged the inclusions into the SENs, proving to be disastrous regarding steel quality [62]. The inclusion trajectories and removal are quantified during the inlet temperature change. However, this study is conducted on a tundish without any internal furniture to optimize its flow and inclusion removal, and the maximum inclusion size simulated is only 150μm.

In this study, a three-dimensional non-isothermal CFD model is developed, for the first time, to investigate the effect of decreasing inlet temperature by 20 K on the transient behavior of fluid flow in a two-strand caster tundish with internal furniture (impact box, weir and dam). The removal of inclusions is quantified by its exit fraction at tundish outlet. The material properties and casting conditions are listed in

Table 7.1. The inclusion density is set as $5000kg/m^3$ by assuming that the inclusion is half Al_2O_3 and half steel [120]. Superheat is defined as the temperature above the steel liquidus temperature, $T_{steel,liq}, 1796K$..

Table 7.1 Caster Dimensions and Process Conditions

Parameters	Symbols	Value
Tundish liquid height (mm)	h_t	1200
Steel flow rate (metric ton/min)	m	1.45
Casting speed (m/s)	v_c	0.01
Molten steel density (kg/m^3)	ρ	7000
Molten steel dynamic viscosity (kg/m·s)	μ_0	0.0063
Liquidus Temperature (K)	$T_{steel,liq}$	1796
Inlet superheat (K)	$\triangle T_{sup,in}$	30 for steady-state simulation 10 for transient simulation
Steel heat capacity (J/kg·K)	c_p	680 [13]
Steel thermal conductivity (W/m·K)	k_0	26 [13]
Steel thermal expansion coefficient (K^{-1})	β	1.0x10$^-$4 [13]
Inclusion density (kg/m^3)	ρ_p	5000
Inclusion heat capacity (J/kg·K)	$c_{p,p}$	718

7.3 Computational Models and Solution Procedure

In this study, a steady-state three-dimensional single-phase CFD model with standard k-ϵ model is developed first to simulate the steel flow in the tundish with 30K-superheat inlet temperature(T_{in}=1826K) and standard k-ϵ model. Then, the steady-state velocity, turbulent and temperature fields are used as initial conditions for the transient simulation where the inlet temperature is decreased to 1806K at t=0s. Inclusions are injected and tracked by Lagrangian Discrete Phase Model (DPM) with RWM in a one-way-coupling way, because the total oxygen mass concentration is only 15 ppm from the plant measurements, and the effect of inclusions on steel flow is negligible. The models are implemented with the commercial package ANSYS FLUENT [67].

7.3.1 Governing Equations for Single-Phase Flow

For the single-phase iso-thermal steel flow, a mass balance and three momentum balance equations are solved throughout the tundish domain as follows:

$$\frac{\partial}{\partial x_i} u_{is} = 0 \tag{7.1}$$

$$\frac{\partial(\rho_s u_{is})}{\partial t} + \frac{\partial \rho_s u_{js} u_{is}}{\partial x_j} = -\frac{\partial P^*}{\partial x_i} + \frac{\partial}{\partial x_j}[(\mu_{eff})(\frac{\partial u_{is}}{\partial x_j} + \frac{\partial u_{js}}{\partial x_i})] \tag{7.2}$$

where μ_{eff} =$\mu_s + \mu_{ts} = \mu_s + \rho_s C_\mu \frac{k}{\epsilon^2}$. ρ_s is the steel density. P^* is modified pressure ($P^* = P + \frac{2}{3}\rho_s k$), P is static pressure. The effective viscosity, μ_{eff}, is the summation of steel molecular viscosity, μ_s, and turbulent viscosity, μ_{ts}. The latter viscosity depends on a constant C_μ, 0.09, the turbulent kinetic energy, k, and its dissipation rate, ϵ, which can be solved by the two-equation standard $k-\epsilon$ model [95]:

$$\frac{\partial \rho_s k}{\partial t} + \frac{\partial \rho_s u_{is} k}{\partial x_i} = \frac{\partial}{\partial x_j}[(\mu_s + \frac{\mu_{ts}}{\sigma_k})\frac{\partial k}{\partial x_j}] + G_k - \rho_s \epsilon \tag{7.3}$$

$$\frac{\partial \rho_s \epsilon}{\partial t} + \frac{\partial \rho_s u_i \epsilon}{\partial x_i} = \frac{\partial}{\partial x_j}[(\mu_s + \frac{\mu_{ts}}{\sigma_\epsilon})\frac{\partial \epsilon}{\partial x_j}] + \frac{\epsilon}{k}(C_{1\epsilon} G_k - C_{2\epsilon}\rho_s \epsilon) \tag{7.4}$$

where G_k is the generation of k due to mean velocity gradients. σ_k and σ_ϵ are turbulent Prandtl number associated with k and ϵ, 1.0 and 1.3 respectively. $C_{1\epsilon}$ and $C_{2\epsilon}$ are constants, 1.44 and 1.92.

For single-phase heat-transfer flow, the Boussinesq approximation is assumed as the product of thermal expansion coefficient and temperature difference, $\beta \triangle T$ is $<< 0.01$, the continuity equation is the same as Equation 7.1, but the momentum equation becomes:

$$\frac{\partial(\rho u_{is})}{\partial t} + \rho_s u_{js}\frac{\partial u_{is}}{\partial x_j} = -\frac{\partial P^*}{\partial x_i} + \frac{\partial}{\partial x_j}[(\mu_{eff})(\frac{\partial u_{is}}{\partial x_j} + \frac{\partial u_{js}}{\partial x_i})] - \rho_0 \beta (T - T_0) g \tag{7.5}$$

where T is the local predicted temperature (K), T_0 is the reference temperature (K), g is the gravitational acceleration, and β is the thermal expansion coefficient (K^{-1}). ρ_0 is the fluid density under T_0.

The energy equation is:

$$\rho_s u_{is}\frac{\partial h_s}{\partial x_i} = \frac{\partial}{\partial x_i}(k_s + k_{ts})\frac{\partial T}{\partial x_i} \tag{7.6}$$

where h_s is the sensible energy, k_s is the fluid heat conductivity, k_{ts} is the turbulent heat conductivity and is calculated by:

$$k_{ts} = \frac{c_{ps}\mu_{ts}}{P_{rt}} \tag{7.7}$$

where c_{ps} is the heat capacity, P_{rt} is the turbulent Prandtl number, 0.85.

7.3.2 Lagrangian DPM Model

The Lagrangian DPM Model is used to simulate inclusions transport in the molten steel pool in the tundish in a one-way-coupled way.

The following momentum balance equation is solved for each inclusion with velocity u_{ip}:

$$\frac{Du_{ip}}{Dt} = \underbrace{\frac{18\mu}{\rho_p d_p^2}\frac{C_{D2}}{24}(u_{is}-u_{ip})\frac{\overbrace{\rho_s d_p \mid u_{ip}-u_{is} \mid}^{Re_p}}{\mu}}_{F_{iD}} + \underbrace{0.5\frac{\rho_s}{\rho_p}(\frac{Du_{is}}{Dt}-\frac{Du_{ip}}{Dt})}_{F_{iVM}} + \underbrace{\frac{\rho_s}{\rho_p}u_{ip}\frac{\partial u_{is}}{\partial x}}_{F_{iP}} + \underbrace{\frac{g(\rho_p-\rho_s)}{\rho_p}}_{F_{iB}} \tag{7.8}$$

where F_{iVM} is virtual mass force, per unit inclusion mass an additional force required to accelerate the surrounding fluid when the inclusion is accelerated. F_{iP} and F_{iB} are pressure gradient force and buoyancy/gravity force per unit inclusion mass. Re_p is the particle Reynolds number. F_{iD} is the drag force per unit inclusion mass, and the drag coefficient, C_{D2}, in DPM model was computed based on the drag law proposed by Moris *et.al*[98].

$$C_{D2} = a_1 + \frac{a_2}{Re_p} + \frac{a_3}{Re_p^2} \tag{7.9}$$

$$a_1, a_2, a_3 = \begin{cases} 0, 24, 0 & \\ 3.690, 22.73, 0.0903 & 0.1 < Re_p < 1 \\ 1.222, 29.1667, -3.8889 & 1 < Re_p < 10 \\ 0.6167, 46.50, -116.67 & 10 < Re_p < 100 \\ 0.3644, 98.33, -2778 & 100 < Re_p < 1000 \\ 0.357, 148.62, -47500 & 1000 < Re_p < 5000 \\ 0.46, -490.546, 578700 & 5000 < Re_p < 10000 \\ 0.5191, -1662.5, 5416700 & Re_p > 10000 \end{cases} \tag{7.10}$$

When one-way coupled with the single-phase RANS model, a RWM is used to account for the effect of small eddies on the inclusions. The isotropic RWM is used to approximate the chaotic dispersion of particles due to turbulence. In this model, a Gaussian distributed random velocity fluctuation, u_i' is generated using Equation (24), which assumes isotropic turbulence everywhere in the fluid domain, in proportion to the local turbulence level.

$$u_i' = \xi\sqrt{2k/3} \tag{7.11}$$

where ξ is a standard normally-distributed random number (i.e. mean = 0 and standard deviation = 1). The random number is not changed until the interaction time of the eddy and a particle t_{inter} first reached the eddy life-time t_e or the eddy cross time t_{cross} (the time required for a particle to pass the eddy). These two time scales are defined as:

$$t_e = -0.15\frac{k}{\epsilon}\ln\gamma \tag{7.12}$$

$$t_{cross} = -\tau_p \ln[1 - \frac{L_e}{\tau_p \mid u_{is} - u_{ip} \mid}] \tag{7.13}$$

where γ is a uniformly-distributed random number from 0 to 1. This randomness of γ generates a random eddy life time that increases in regions with high local turbulence and little dissipation. L_e is the eddy length scale.

7.4 Flow Model Domain, Mesh, and Boundary Conditions

Due to two-fold symmetry, only a quarter of the tundish is modeled to cut down the computational time. The flow model domain, including the impact box, dam, weir and liquid pool in the tundish, are shown in Figure 7.1(a). The weir and dam walls divide the tundish into two chambers, the pouring chamber where molten steel is tapped from ladle into the tundish and hit the impact box, and the flotation chamber where the molten steel flows upwards through both the space between weir and dam and the dam hole, and exits the tundish at the bottom outlet. The mesh is created in ANSYS Workbench and includes ~0.36 million all-hexahedral cells with mesh sensitivity study, as shown in Figure 7.1(b). The mesh is refined near the impact box, dam hole and outlet due to high velocity gradient in these regions, as shown in Figure 7.1 (c) and (d). The cell volume ranges from $6.4 \times 10^{-}8$ to $1.6 \times 10^{-}5 m^3$, which corresponds to a cell length from 4 to 25mm. The Y^+ value varies from 0.5 to 200 across the tundish walls, with maximum Y^+ occurring at the jet impingement region of the impact box, so the standard wall function is used.

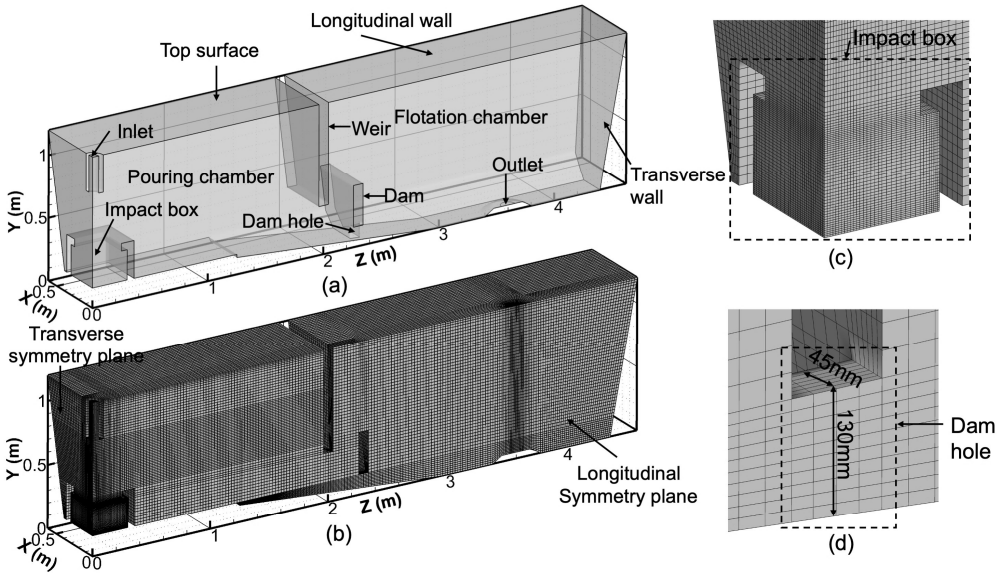


Figure 7.1 (a) Geometry of the whole domain (b) mesh of the whole domain (c) mesh near impact box and (d) mesh near dam hole

For the steel-phase flow simulation, constant mass flow rate 24.15 kg/s is given at the inlet. with 5% turbulent intensity and 0.105 m hydraulic diameter. Steel is entering in a direction normal to the inlet. The top surface is approximated as non-slip for the molten steel flow, due to the large viscosity of the slag layer. Other walls (longitudinal wall, transverse wall, impact box wall, weir wall and dam walls) are also set as non-slip walls. A pressure outlet condition is chosen over the outlet, according to the ferrostatic pressure head of molten steel ($\rho_s g h_m$), where h_m is the distance below the meniscus. The outlet pressure is calculated by a UDF in Appendix A. Reverse flow has 5% turbulent intensity and 0.08m hydraulic diameter. All variable gradients on the symmetry planes are zero. Standard wall function is used for boundary-layer velocity approximation for the wall-adjacent cells.

Heat transfer boundary conditions are fixed heat fluxes taken from Chakraborty and Sahai [60], as summarized in Table 7.2. The impact box is adiabatic because there's no place for the heat to escape. The inlet temperature is 1826 K (30 K superheat)) and 1806 K (10 K superheat) for steady-state and transient simulation, respectively. Reverse-flow temperature is assumed to be 1796 K (the steel liquidus temperature).

Table 7.2 Thermal Boundary Conditions

Parameters	Symbols	Value
Top surface (kW/m^2)	q_{top}	15
Bottom surface (kW/m^2)	q_{bottom}	1.4
Longitudinal wall (kW/m^2)	q_{long}	3.2
Transverse wall (kW/m^2)	q_{tran}	3.8
Weir and dam wall (kW/m^2)	q_{wd}	1.75

100,000 inclusions are injected into the steady-state flow once and for all at t=0 s, with a size distribution of 10,000 inclusions for each of the 9 diameters ($10\mu m$, $20\mu m$, $30\mu m$, $50\mu m$, $100\mu m$, $200\mu m$, $300\mu m$, $500\mu m$, $1{,}000\mu m$). For the transient flow, a burst of 10,000 inclusions is injected for each of the 9 inclusion diameters every 5 s since t=0 s. The injected inclusions can either escape into the top surface or exit the outlet. The reflection condition is employed on other walls.

For inclusions injected into steady-state flow, the time step size for inclusion trajectory calculation is 0.005 s. For transient simulation, the time step size for both fluid flow and inclusion track is 0.005 s, since they are solved simultaneously.

A Semi-Implicit Pressure Linked Equations (SIMPLE) algorithm is used to solve the equation set of one mass balance, three momentum balance, one energy balance, and the two standard k-ϵ turbulent equations with second-order upwind discretization scheme for convective terms. Second order implicit method is used for temporal term discretization. Pressure Staggering Option (PRESTO!) scheme is used for pressure

interpolation by using stagger grids to store pressure values at cell center and interpolate pressure gradients on cell face. The under-relaxation factors are set as 0.1 for pressure, 0.2 for momentum and turbulence, and 0.8 for energy. Convergence is reached when all residuals reach 1^{-6}. The steady-state three-dimensional flow requires 20,000 iterations and ~7 hours of CPU time with 10 computational nodes on a Dell computer tower with Intel(R) Xeon(R) E5-2609 v3 @1.90GHz CPU and 64.0GB RAM. The transient simulation uses the steady-state results as initial conditions at t=0 s with lower inlet temperature (1806K), and it requires about 4 months of computational time to get 4,000 s flow time on high-performance computer Mio with 48 computational nodes. The limited computational power is the main reason why a quarter domain with standard k-ϵ model are adopted, as they allow a fewer-cell and coarser mesh.

7.5 Steady-State Temperature Distribution Results

Figure 7.2 shows the temperature distribution on the symmetry planes and walls. It appears that the temperature gradually goes down with the steel marching towards the outlet due to heat loss into surrounding walls. In most of the domain, the temperature goes down within 10K from its inlet temperature, 1826K, except the far-end region of top surface where heat loss happens in three directions simultaneously: top surface, longitudinal wall, and transverse wall, and the lowest temperature there is 1810K. This 16K (1826K - 1810K) lower temperature indicates that lower superheat may lead to top surface freeze near the top surface corners.

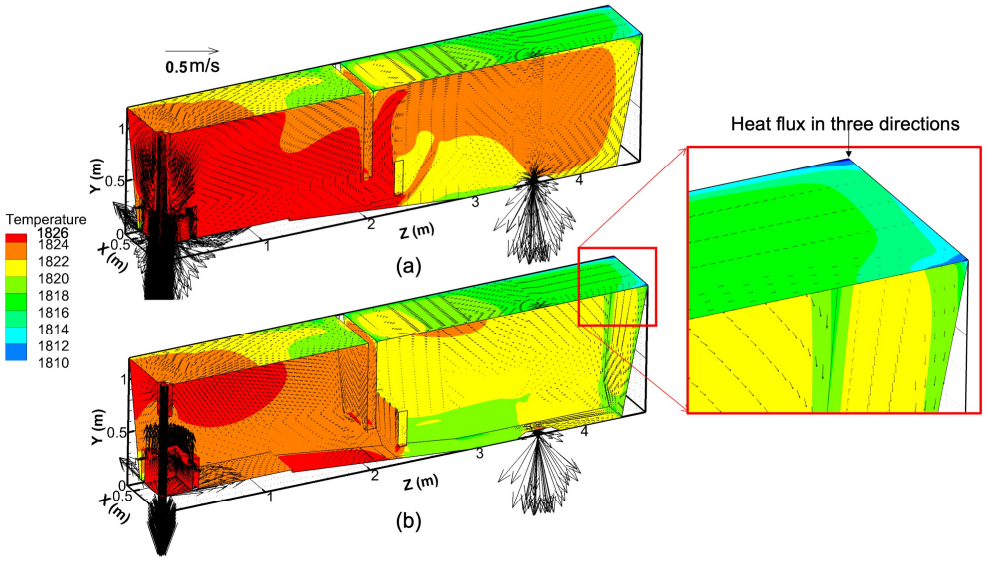


Figure 7.2 Steady-state temperature distribution on (a) symmetry planes, top surface, weir wall, and dam wall, and (b) top surface, impact box wall, weir wall, dam wall, longitudinal wall, and transverse wall

The outlet temperature is 1822K, only 4 K lower than the inlet temperature, as shown in Figure 7.3. Only 0.3% enthalpy is conducted into the surrounding walls, and half of the heat loss happens on top surface due to its large heat flux and area.

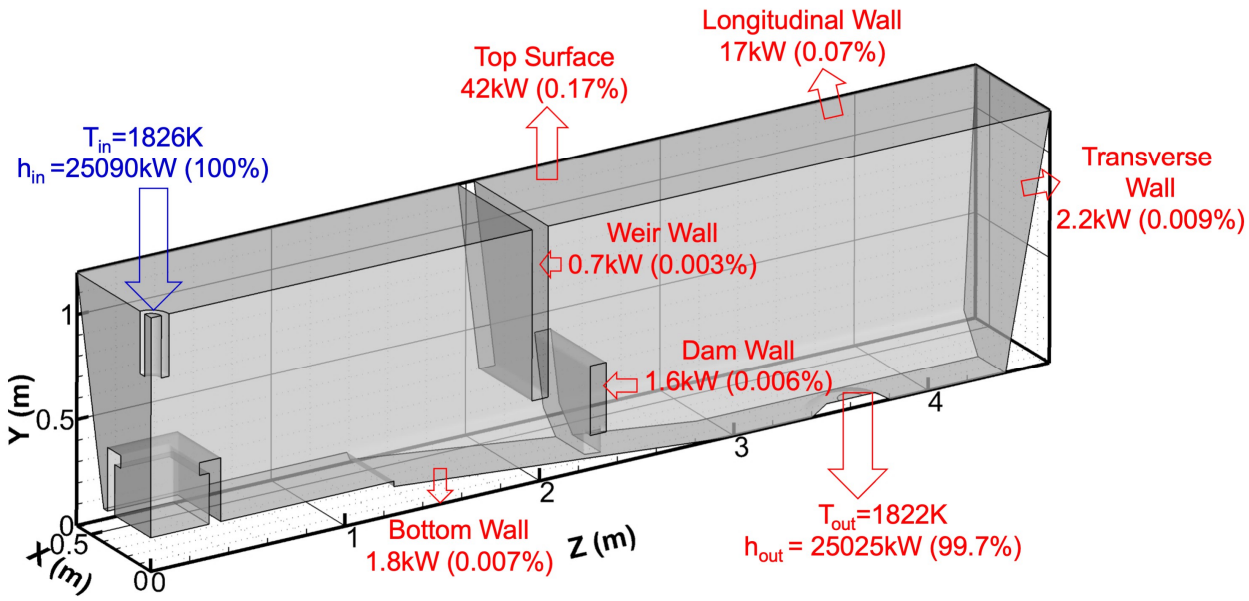


Figure 7.3 Flow pattern on center plane at (a) t=40s, (b) t=50s, (c) t=60s, (d) t=70s

7.6 Transient Flow Results- Steel Velocity and Temperature

Figure 7.4 (see page 112) shows the steel velocity magnitude distribution and vector on the longitudinal symmetry plane at t=0s, 100s, 200s, 1,000s, 1,700s, and 2,700s. The steady-state results are used as initial conditions at t=0s, as shown by Figure 7.4(a), and the inlet temperature is decreased from 1826K (30K superheat) to 1806K (10K superheat) since t=0s. At t=100s, jet 1 and jet 3 go down to the bottom wall, because of the colder steel (10K superheat) tends to flow downwards and bring the jets down. At 200s, all three jets are brought down by the colder steel to flow along the bottom floor of tundish and much more inclusions will exit the outlet during this period, because of the short-cut path, until the jets turn back up. At 1,000s, jet 1 fully resumes its steady-state flow pattern, but jet 2 and 3 stay downwards. At 1,700s, jet 2 starts to turn upwards and turns fully back up at 2,700s. The steel flow at 2700s is similar to steady state (t=0s), meaning that all the hot steel is replaced by the cold steel and the flow reaches quasi-steady state. The inclusion exit fraction at outlet after 2700s should be similar to that under steady state. To quantify the jet-downwards period and its effect on inclusion exit fraction at outlet, z velocity (V_z) are monitored at five points: P1t, P1b, P2t, P2b, P3, as shown in Figure 7.4.

Figure 7.5 (see page 112) shows the steel temperature distribution and velocity vector on the longitudinal symmetry plane at t=0s, 100s, 200s, 1,000s, 1,700s, and 2,700s. Cold steel flows to the tundish bottom immediately after coming in, and a stratified temperature profile occurs. In Figure 7.5(e), the temperature in flotation chamber has been lower then 1806K, and that's when the jet 2 starts to turn back up in Figure 7.4(e). At 2,700s, the temperature distribution reaches quasi-steady state.

The steel residence time in the tundish, t_R, can be estimated by the equation below:

$$t_R = \frac{V_t \rho}{\dot{m}_{in}} \tag{7.14}$$

where V_t is the tundish liquid-pool volume, 2.44m^3, $\dot{m}_{in}$ is the inlet steel flow rate, 24.15kg/s. Then, t_R=707s, meaning it takes 707s for the new-feeding steel to replace all old steel in the tundish. However, in this study, it takes 2,700s for the cold steel to replace all the hot steel, which is ∼4× longer than the calculation, because that cold steel tends to flow along the tundish floor, forming a stratified flow and temperature profile until the hot steel is cooled down and the cold steel stream turns upwards. This phenomenon apparently delays the feeding and mixing process.

Figure 7.6 (see page 113) shows the monitored Z velocity at five points on the longitudinal symmetry plane. P1t and P1b are two points near jet 1 and near the top and bottom wall respectively. First, the velocity at P1t goes all the way down from ∼0.064m/s to ∼-0.038m/s within the first 100s (black solid line), meanwhile the velocity at P1b increases significantly from ∼-0.014m/s to ∼0.09m/s. This shift in velocity means that jet 1 is brought down by the cold steel, which decreases the near-top-surface velocity but increases the near-bottom-wall velocity, as shown by the black downward arrow in Figure 7.6. At ∼400s, jet 1 starts to turn back up, as the velocity at P1b starts to drop and that at P1t begins rising, as shown by the black upward arrow. Jet 1 turns fully back up at ∼1,000s. Similarly, jet 2 starts going down at ∼150s and turning up at ∼700s, and it gets fully back up at ∼1,800s. Finally, jet 3 goes down at ∼100s and very slowly turns up.

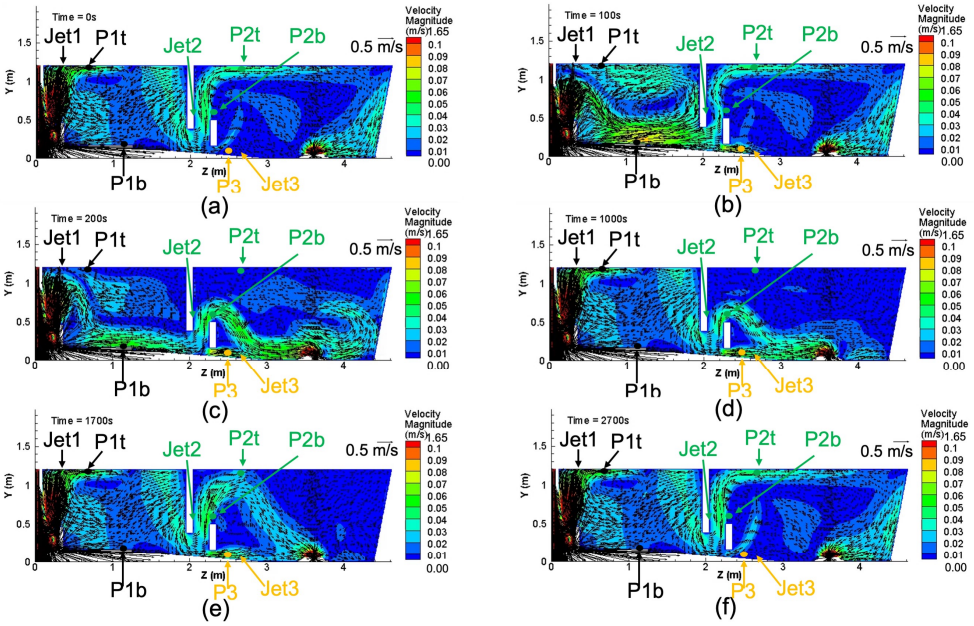


Figure 7.4 Steel velocity magnitude distribution and vector on longitudinal symmetry plane at t = (a) 0s, (b) 100s, (c) 200s, (d) 1,000s, (e) 1,700s, and (f) 2,700s

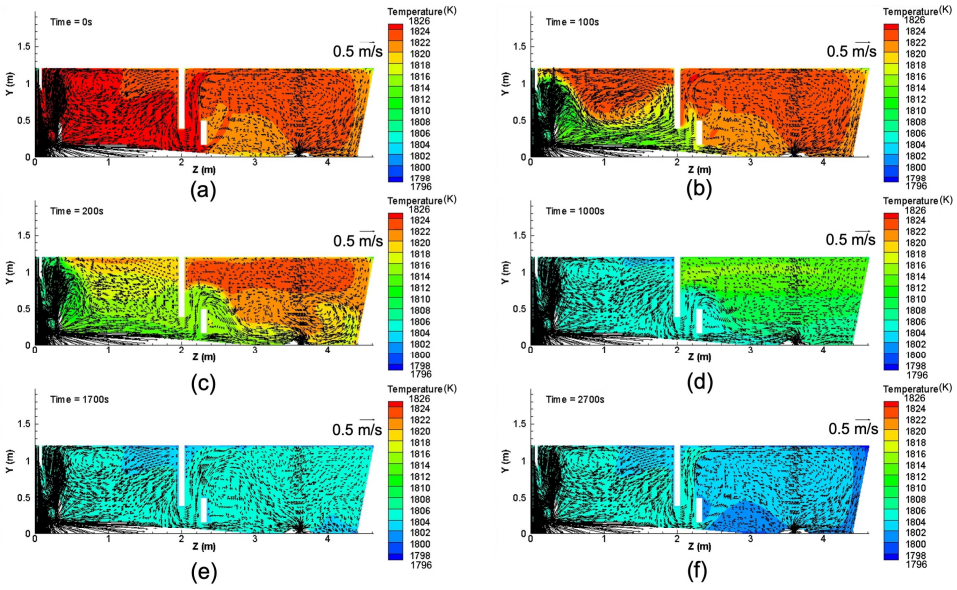


Figure 7.5 Steel temperature distribution and velocity vector on longitudinal symmetry plane at t = (a) 0s, (b) 100s, (c) 200s, (d) 1,000s, (e) 1,700s, and (f) 2,700s

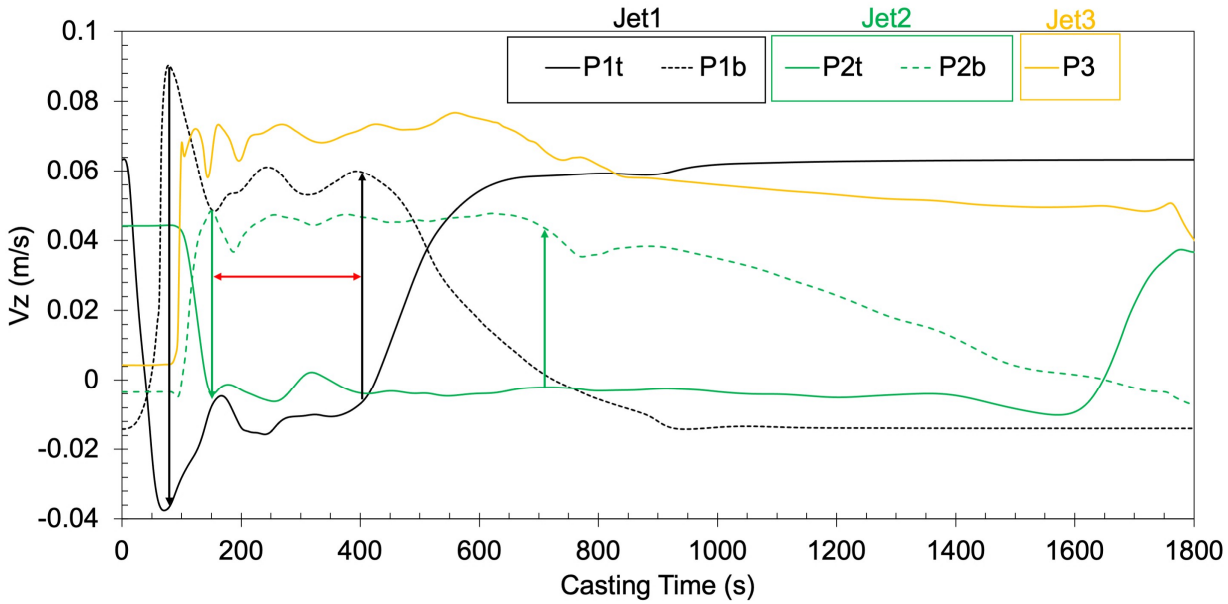


Figure 7.6 Z velocity, V_z, evolution for 1,800s flow time at five monitored points, P1t, P1b, P2t, P2b, and P3

From 150s to 400s, all the three jets are fully brought down, as shown by the red horizontal double-sided arrow between the green downward arrow and the black upward arrow, and the flow pattern within this period is shown in Figure 7.4(c), where the inclusion's path is directly short-cut to the outlet, leading to maximum exit fraction and worst steel quality.

7.7 Inclusion Transport and Exit Fraction Results

To quantify the effect of flow pattern on inclusion, the definition of inclusion exit fraction,$C_{f,i}$, for inclusions with diameter d_{pi} and injected at any time point, is proposed in this work as:

$$C_{f,i} = \frac{N_{out,i}}{N_{in,i}} \tag{7.15}$$

where $N_{out,i}$ is the number of inclusions with diameter d_{pi} exiting tundish outlet after $N_{out,i}$ reaches constant, and $N_{in,i}$ is the number of inclusion injected with diameter d_{pi} at the inlet.

Figure 7.7 (see page 115) shows how many of the 10μm inclusions injected at 100s, 500s and 1,000s after the switch of ladle temperature exiting tundish outlet. For all the three lines, very small portion of inclusions exit outlet within the first 100s after injection, because it takes time to transport them through the tundish. 100s after injection, the number of inclusions exiting outlet starts to increase, shown by the steep slopes. First two lines gradually get flat ~2,000s after injection, meaning $N_{out,i}$ reaches constant and the exit fraction can be calculated using equation (7.15). later being injected, smaller the number of

inclusions exiting outlet due to the turning up of the three jets.

Similarly, Figure 7.8 (see page 115) shows the number of 100μm inclusions exiting outlet. Same conclusions hold true in Figure 7.8. However, $N_{out,i}$ reaches constant $\sim$ 900s after injection, which is only half of that of 10μm inclusions. This is because large inclusions tend to flow upwards into the slag layer due to larger buoyancy force and stoke rising velocity, thus there's fewer large inclusions to track and it is easier for them to reach steady state.

Figure 7.9 (see page 116) shows the exit fraction, $C_{f,i}$, for inclusions smaller than and equal to 100μm injected within the first 2500s. The exit fraction is calculated with $N_{out,i}$ at 4,000s flow time when all the $N_{out,i}$ for inclusions injected within the first 2,500s reach constant. First, the largest $C_{f,i}$ occurs for inclusions injected at 150s. Inclusions injected from 50s to 300s has largest exit fraction, because they go through the flow with three jets all going down as shown by the red arrow in Figure 7.6. Smaller the inclusions, higher the exit fraction, because the smaller inclusions tend to go with the flow instead of flowing upwards into slag layer. At 150s, $C_{f,i}$ decreases from $\sim$80% to $\sim$35% with inclusion diameter increasing from 10μm to 100μm.

In Figure 7.9, taking 10μm inclusions for example, $C_{f,i}$ initially rises up from $\sim$40% at steady state to about 80% for inclusions injected at 150s, because the going down of jet 1, 2 and 3. Then, $C_{f,i}$ gradually drops to about 60%, because jet 1 turns back up. For inclusions injected from 550s to 1350s, the $C_{f,i}$ stays constant, because jet 2 and 3 stays down during that period of time. For inclusions injected from 1350s to 1900s, its exit fraction gradually reduces from about 60% to about 40% due to the turning up of jet 2 and 3. For inclusions injected after 1900s, its exit fraction is close to $\sim$40%, which is the exit fraction under steady state.

Figure 7.10 (see page 116) shows the exit fraction, $C_{f,i}$, for inclusions larger than 100μm injected every 100s interval within the first 2,000s flow time. Not like in Figure 7.9, where $C_{f,i}$ drops due to jet 1 turning upwards for inclusions injected after 150s, exit fraction for 200μm and 300μm inclusions stays constant until 1500s of injection, as a results of dominant effect of jet 2 and 3 on the exit fraction of large inclusions. Jet 1's effect on inclusions larger than 100μm is negligible, because jet 1 flows fast at 1.65m/s and a fixed portion of large inclusions (>100μm) tend to flow into top slag layer due to large inertia effect and buoyancy force. Jet 2 and 3 dominates over jet 1 in terms of exit fraction for large inclusions (>100μm) because of lower velocity/inertia effect and being deeper below the top surface.

Combining Figure 7.9 and Figure 7.10, the exit fraction gets worse for the first 1900s ($\sim$32min) flow time, corresponding to a 1900s$\times$0.01m/s = 19-m-long slab. Switching hotter ladle by colder ladle can possibly lead to more inclusion defects and worse steel quality in a 19-m-long slab, and should be avoided by pre-heating the ladle to the same temperature.

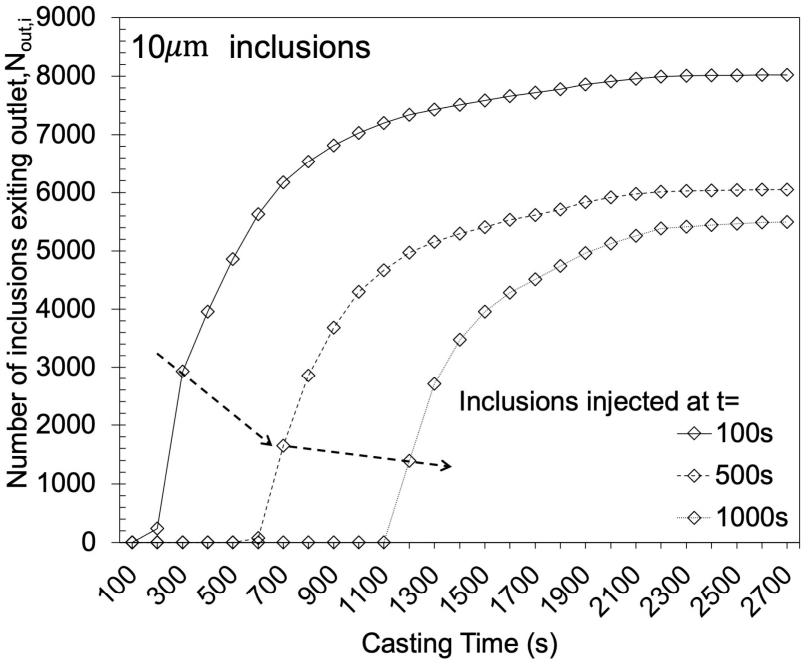


Figure 7.7 Number of 10μm inclusions exiting outlet vs. casting time

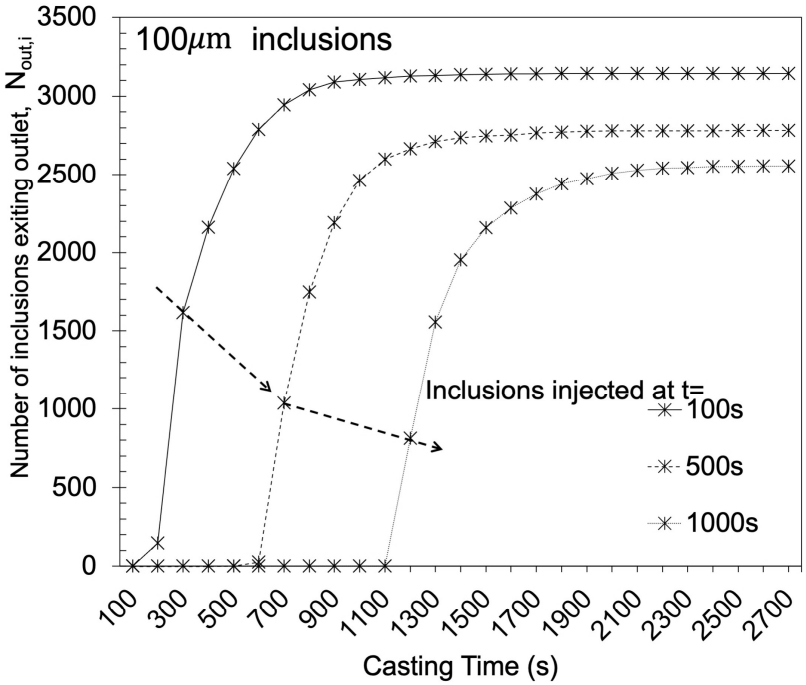


Figure 7.8 Number of 100μm inclusions exiting outlet vs. casting time

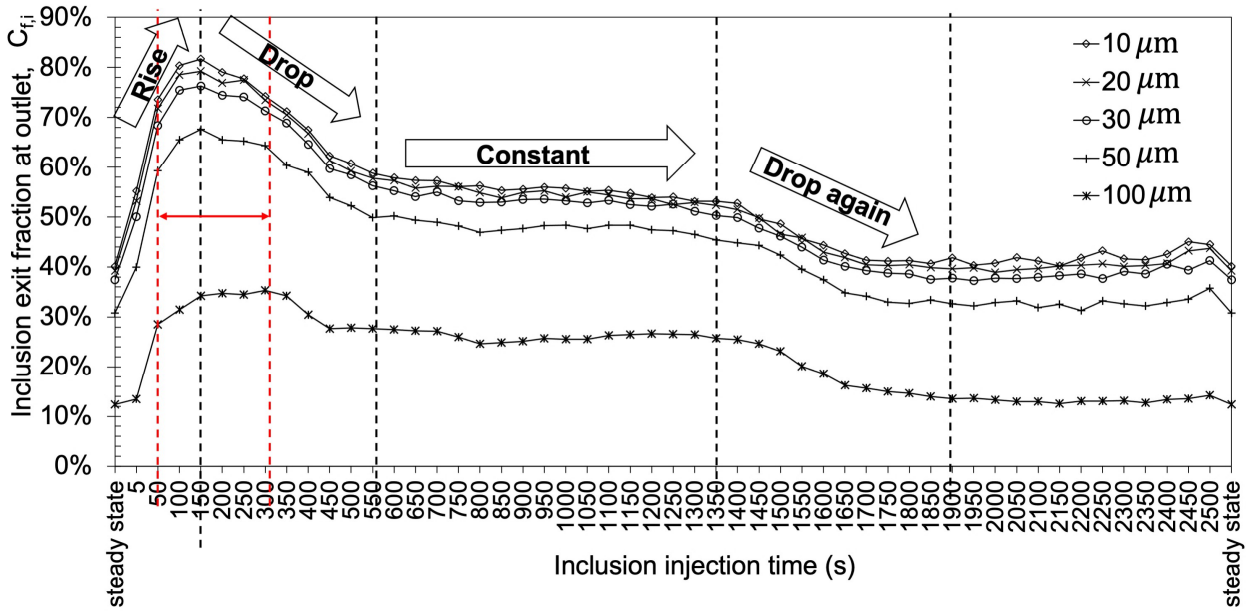


Figure 7.9 Exit fraction, $C_{f,i}$, for inclusions smaller than and equal to 100μm injected within the first 2500s flow time

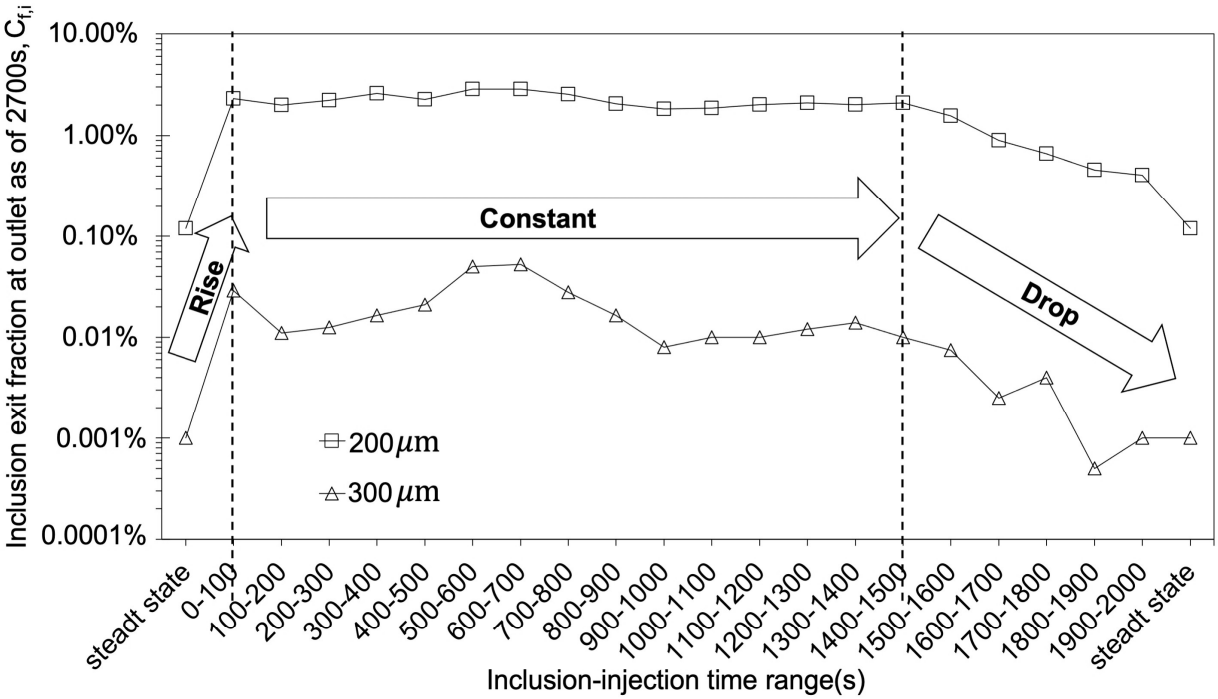


Figure 7.10 Exit fraction, $C_{f,i}$, for inclusions larger than 100μm injected every 100s interval within the first 2,000s flow time

Figure 7.11 compare the exit fraction under steady state and the maximum exit fraction when decreasing the inlet superheat by 20K (1826K - 1806K = 20K). For small inclusions ($\leq$100μm), the exit fraction is 2-3$\times$ larger by decreasing the inlet temperature. For large inclusions (>100μm), decreasing inlet superheat by 20K increases the exit fraction by 32$\times$ and 80$\times$ for 200μm and 300μm inclusions, respectively. No 500μm and 1,000μm inclusions exit the tundish outlet.

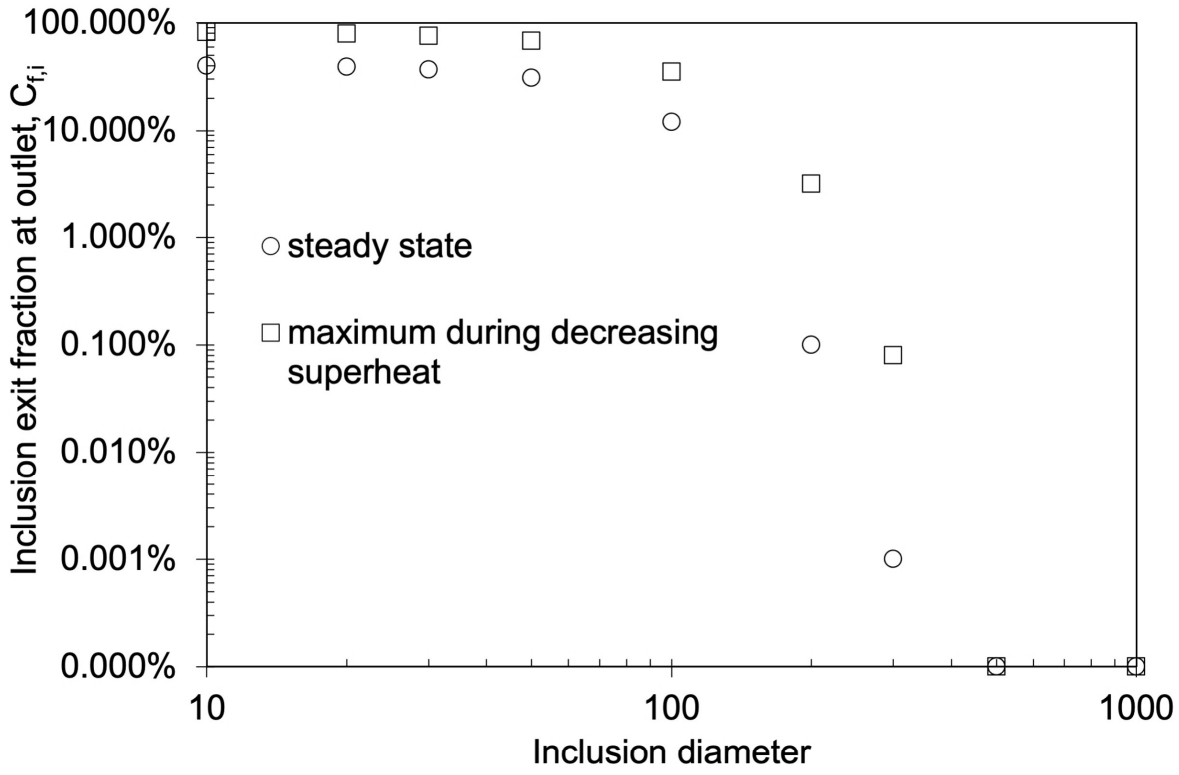


Figure 7.11 Exit fraction of different inclusions

7.8 Conclusions

A three-dimension CFD model is developed to investigate the effect of decreasing inlet superheat by 20K on tundish flow pattern and inclusion exit fraction.

It is reasonable to assume that the effect of keeping ladle superheat the same, i.e. pouring 30 K / 10 K-superheat steel into 30 K / 10 K-superheat steel respectively, has negligible effect on its flow pattern and inclusion exit fraction; effect of increasing ladle superheat, i.e. pouring 30K superheat steel into 10K superheat steel also has trivial effect, because hot steel tends to float up and likely leads to slightly lower inclusion exit fraction.

However, pouring cold steel (10K superheat) into hot steel (30K superheat) has significant effect on flow pattern:

1. Cold steel tends to flow downwards and brings the three jets down, leading to more inclusion exiting outlet.

2. Both velocity and temperature distribution reach quasi-steady state after 2,700s of decreasing inlet temperature, which is about 4× longer than the regular steel residence time (707s) due to jets stay down, forming a stratified flow.

For small inclusions ($\leq$100μm):

1. Largest exit fraction occurs for inclusions ($\leq$100μm) injected between 50s – 300s.

2. Smaller the inclusion, higher the exit fraction because smaller particles are easier to flow with the jet.

3. Initial rise of exit fraction after decreasing inlet temperature by 20K is due to Jet1, 2 and 3 going downwards; The first drop in exit fraction is due to Jet1 going back up; Constant exit fraction is due to jet 2 and 3 maintaining downwards for a long period of time; finally, the second drop of exit fraction is due to jet 2 and 3 turning back up

4. For 10 μm inclusions, the exit fraction increases ×2 times; while for 100 μm inclusions, it increases ×3 times.

For larger inclusions (>100μm):

1. Jet 1's going down has negligible effect on inclusions (>100μm) because of its high velocity/inertia effect towards top surface and large inclusions' larger buoyancy force.

2. Jet 2 and 3 dominate over jet 1 in terms of inclusions (>100μm) exit fraction, which is reflected as the almost constant exit fraction for inclusions injected before 1500s.

3. The exit fraction decreases for inclusions injected after 1500s because Jet 2 starts to turn upwards at 1600s.

4. Decreasing ladle temperature by 20K can make the exit fraction of 200μm and 300μm inclusions 32 and 80 times larger than steady-state happening when feeding the same-temperature steel or hotter steel).

The increase of inclusions exiting the tundish outlet may cancel off the less and shallower capture from higher superheat (as shown in Chapter 4), if the successive ladle has lower temperature.

CHAPTER 8

MODELING OF STEEL FLOW, SUPERHEAT, IMPACT BOX MISALIGNMENT, INCLUSION TRANSPORT, AND REMOVAL IN A TWO-STRAND TUNDISH

8.1 Introduction

Tundish serves as a buffer zone between the ladle and the mold. The inclusions from upstream ladle refining process flow into the tundish. Part of them float up into the top slag layer and be permanently removed, the rest exits the tundish and flows into the mold. If they get captured in the strand, permanent surface and internal defects form in the final product.

As shown in Chapter 4, superheat can change the lower strand flow pattern and lower the number of particles captured. However, the effect of superheat on a steady-state tundish flow and particle exit fraction at the tundish outlet has little investigation. Other than superheat, it is easy to have misalignment between tundish insertion nozzle and the bottom impact box, which leads to asymmetrical flow pattern and more particles exiting the tundish outlet on one side than the other.

In this chapter, a three-dimensional steady-state RANS CFD model with SST k-ω turbulence model is developed for a steady-state tundish flow. Inclusions are injected and tracked by the DPM model with RWM model. The effects of superheat and insertion nozzle misalignment on flow pattern and inclusion exit fraction are investigated.

8.2 Computational Models and Solution Procedures

To start with, an iso-thermal single-phase RANS model and SST k-ω turbulence model are used for the steady-state single-phase steel flow pattern for both the symmetrical and nozzle-misaligned domain. Then superheat-flow-coupled (10 K and 30 K) calculation starts with the iso-thermal solution as the initial condition, for the symmetrical computational domain case. Inclusions are injected and tracked by a DPM model, they can either be removed by the top slag layer or exit the two-strand tundish outlets. Because the total oxygen mass concentration is only 15 ppm, the inclusions are tracked in a one-way-coupled way. The second order upwind scheme is used for momentum, k and ω equations. PRESTO! is used for pressure scheme (staggered grid).

8.2.1 Governing Equations for Single-Phase Flow

For the single-phase iso-thermal steel flow, a mass balance and three momentum balance equations are solved throughout the tundish domain:

$$\frac{\partial}{\partial x_i}u_{is} = 0 \tag{8.1}$$

$$\frac{\partial(\rho u_{is})}{\partial t} + \frac{\partial \rho_s u_{js} u_{is}}{\partial x_j} = -\frac{\partial P^*}{\partial x_i} + \frac{\partial}{\partial x_j}[(\mu_{eff})(\frac{\partial u_{is}}{\partial x_j} + \frac{\partial u_{js}}{\partial x_i})] \tag{8.2}$$

where μ_{eff} =$\mu_s + \mu_{ts} = \mu_s + \rho_s C_\mu \frac{k}{\epsilon^2}$. ρ_s is the steel density. P^* is modified pressure ($P^* = P + \frac{2}{3}\rho_s k$), P is static pressure. The effective viscosity, μ_{eff}, is the summation of steel molecular viscosity, μ_s, and turbulent viscosity, μ_{ts}. The latter viscosity can be solved by the two-equation SST $k-\omega$ model. For single phase flow, the SST k-ω model [121, 122] equations are as follows:

$$\frac{\partial \rho_s k}{\partial t} + \frac{\rho_s u_{is} k}{\partial x_i} = \frac{\partial}{\partial x_i}[(\mu_s + \sigma_{k1}\mu_{ts})\frac{\partial k}{\partial x_i}] - \beta^* \rho_m k\omega + \tilde{P}_k \tag{8.3}$$

$$\frac{\partial \rho_s \omega}{\partial t} + \frac{\rho_s u_{is} \omega}{\partial x_i} = \frac{\partial}{\partial x_i}[(\mu_m + \sigma_\omega \mu_{ts})\frac{\partial \omega}{\partial x_i}] + \alpha_\omega \rho_s S^2 - \beta \rho_s \omega^2 + 2(1-F_1)\rho_s \sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i} \tag{8.4}$$

where μ_t is calculated by

$$\mu_t = \frac{\rho k}{\omega}\frac{1}{max[\frac{1}{\alpha^*}, \frac{SF_2}{\alpha_1 \omega}]} \tag{8.5}$$

For single-phase heat-transfer flow, the Boussinesq approximation is assumed as the product of thermal expansion coefficient and temperature difference, $\beta \triangle T$ is $<< 0.01$, the continuity equation is the same as Equation 8.1, but the momentum equation becomes:

$$\frac{\partial(\rho u_{is})}{\partial t} + \rho_s u_{js}\frac{\partial u_{is}}{\partial x_j} = -\frac{\partial P^*}{\partial x_i} + \frac{\partial}{\partial x_j}[(\mu_{eff})(\frac{\partial u_{is}}{\partial x_j} + \frac{\partial u_{js}}{\partial x_i})] - \rho_0 \beta(T - T_0)g \tag{8.6}$$

where T is the local predicted temperature (K), T_0 is the reference temperature (K), g is the gravitational acceleration, and β is the thermal expansion coefficient (K^{-1}). ρ_0 is the fluid density under T_0.

The energy equation is:

$$\rho_s u_{is}\frac{\partial h_s}{\partial x_i} = \frac{\partial}{\partial x_i}(k_s + k_{ts})\frac{\partial T}{\partial x_i} \tag{8.7}$$

where h_s is the sensible energy, k_s is the fluid heat conductivity, k_{ts} is the turbulent heat conductivity and is calculated by:

$$k_{ts} = \frac{c_{ps}\mu_{ts}}{P_{rt}} \tag{8.8}$$

where c_{ps} is the heat capacity, P_{rt} is the turbulent Prantdl number, 0.85.

8.2.2 Lagrangian DPM Model

The Lagrangian DPM Model is used to simulate inclusions transport in the molten steel pool in the tundish in a one-way-coupled way.

The following momentum balance equation is solved for each inclusion with velocity u_{ip}:

$$\frac{Du_{ip}}{Dt} = \underbrace{\frac{18\mu}{\rho_p d_p^2}\frac{C_{D2}}{24}(u_{is}-u_{ip})\overbrace{\frac{\rho_s d_p \mid u_{ip}-u_{is} \mid}{\mu}}^{Re_p}}_{F_{iD}} + \underbrace{0.5\frac{\rho_s}{\rho_p}(\frac{Du_{is}}{Dt}-\frac{Du_{ip}}{Dt})}_{F_{iVM}} + \underbrace{\frac{\rho_s}{\rho_p}u_{ip}\frac{\partial u_{is}}{\partial x}}_{F_{iP}} + \underbrace{\frac{g(\rho_p-\rho_s)}{\rho_p}}_{F_{iB}} \tag{8.9}$$

where F_{iVM} is virtual mass force, per unit inclusion mass an additional force required to accelerate the surrounding fluid when the inclusion is accelerated. F_{iP} and F_{iB} are pressure gradient force and buoyancy/gravity force per unit inclusion mass. Re_p is the particle Reynolds number. F_{iD} is the drag force per unit inclusion mass, and the drag coefficient, C_{D2}, in DPM model was computed based on the drag law proposed by Moris *et.al*[98].

$$C_{D2} = a_1 + \frac{a_2}{Re_p} + \frac{a_3}{Re_p^2} \tag{8.10}$$

$$a_1, a_2, a_3 = \begin{cases} 0, 24, 0 & \\ 3.690, 22.73, 0.0903 & 0.1 < Re_p < 1 \\ 1.222, 29.1667, -3.8889 & 1 < Re_p < 10 \\ 0.6167, 46.50, -116.67 & 10 < Re_p < 100 \\ 0.3644, 98.33, -2778 & 100 < Re_p < 1000 \\ 0.357, 148.62, -47500 & 1000 < Re_p < 5000 \\ 0.46, -490.546, 578700 & 5000 < Re_p < 10000 \\ 0.5191, -1662.5, 5416700 & Re_p > 10000 \end{cases} \tag{8.11}$$

When one-way coupled with the single-phase RANS model, a RWM is used to account for the effect of small eddies on the inclusions. The isotropic RWM is used to approximate the chaotic dispersion of particles due to turbulence. In this model, a Gaussian distributed random velocity fluctuation, u_i' is generated using Equation (24), which assumes isotropic turbulence everywhere in the fluid domain, in proportion to the local turbulence level. The random velocity fluctuation is applied on each inclusion at each time step as if it is kicked towards a random direction by a small force. In this way, the effect of small eddies on inclusion movement is considered into the RANS model.

$$u_i' = \xi\sqrt{2k/3} \tag{8.12}$$

where ξ is a standard normally-distributed random number (i.e. mean = 0 and standard deviation = 1). The random number is not changed until the interaction time of the eddy and a particle t_{inter} firstly reached the eddy life-time t_e or the eddy cross time t_{cross} (the time required for a particle to pass the eddy). These two time scales are defined as:

$$t_e = -0.15\frac{k}{\epsilon}\ln\gamma \tag{8.13}$$

$$t_{cross} = -\tau_p \ln\left[1 - \frac{L_e}{\tau_p \mid u_{is} - u_{ip} \mid}\right] \tag{8.14}$$

where γ is a uniformly-distributed random number from 0 to 1. This randomness of γ generates a random eddy life time that increases in regions with high local turbulence and little dissipation. L_e is the eddy length scale.

8.3 Flow Model Domain and Mesh

Figure 8.1 (see page 123) shows the computational domain for the symmetrical two-strand tundish. Only the liquid pool is modeled. The liquid steel level in the tundish is 1200 mm. The tundish is furnished with a dam, a weir, and an impact box to inhibit turbulence. There's a hole on the dam for the molten steel to go through.

Figure 8.2 (see page 123) shows the mesh for the tundish. ∼3.7 million all-hexahedral cells are generated for the domain. The mesh is refined near the walls, so that the SST k-ω model can behave like k-ω model in the near-wall flow and work as standard k-ϵ model for the bulk flow, which combines the advantages of both k-ω and k-ϵ models and gives a more accurate result. The cell volume ranges from $1.3 \times 10^{-}9$ to $1.9 \times 10^{-}5m^3$, which corresponds to a cell length from 1.1 to 27mm. The Y^+ value varies from 0.04 to 200 across the tundish walls, with most regions with a Y^+ value less than 20.

8.4 Boundary Conditions

The boundary conditions for both fluid flow and particle transport are shown in Figure 8.3 (see page 124). As shown in Figure 8.3 (a) The insertion nozzle has an mass flow inlet of 96.6 kg/s. Outlets are set as pressure outlet with a ferrostatic pressure head $\rho_s gh$. All the walls are treated as non-slip walls. The steel density is 7000 kg/m^3 under iso-thermal condition or under the Buossinesq approximation with heat transfer enabled.

Figure 8.3 (b) shows that particles can either escape into the top slag layer or exit the tundish outlets. The time step size for the DPM model is 0.0005s to meet the Courant number. A burst of 10,000 inclusions are injected at the nozzle inlet for each of the 10 inclusion diameters: 10 μm, 20 μm, 30 μm, 40 μm, 50

μm, 60 μm, 70 μm, 80 μm, 90 μm, 100 μm.

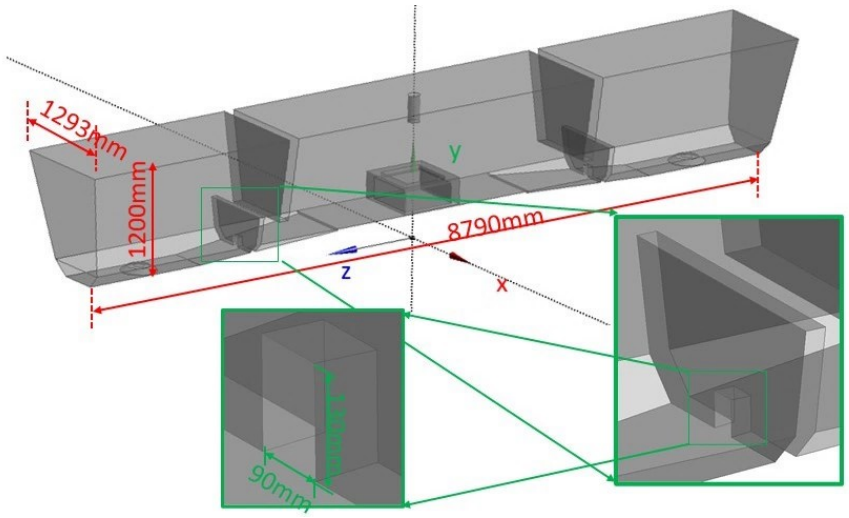


Figure 8.1 Computational domain for the two-strand symmetrical tundish

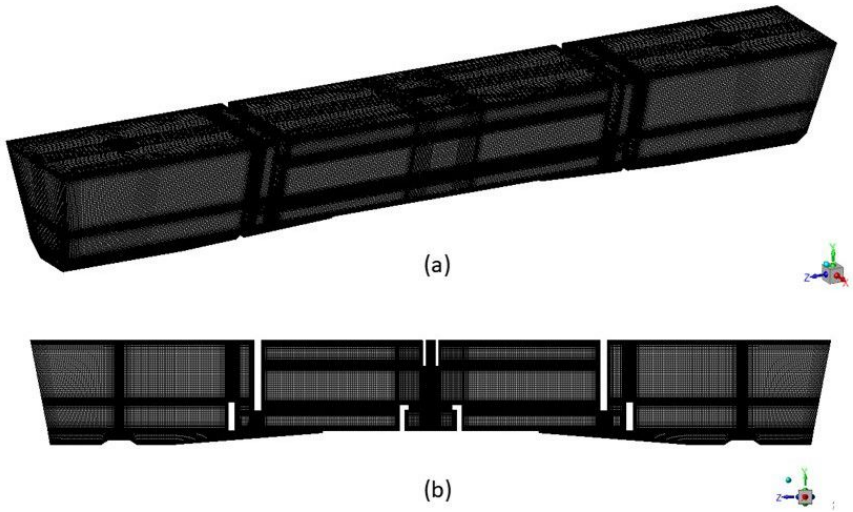


Figure 8.2 Computational mesh for the two-strand symmetrical tundish (a) 3-D view and (b) Front view

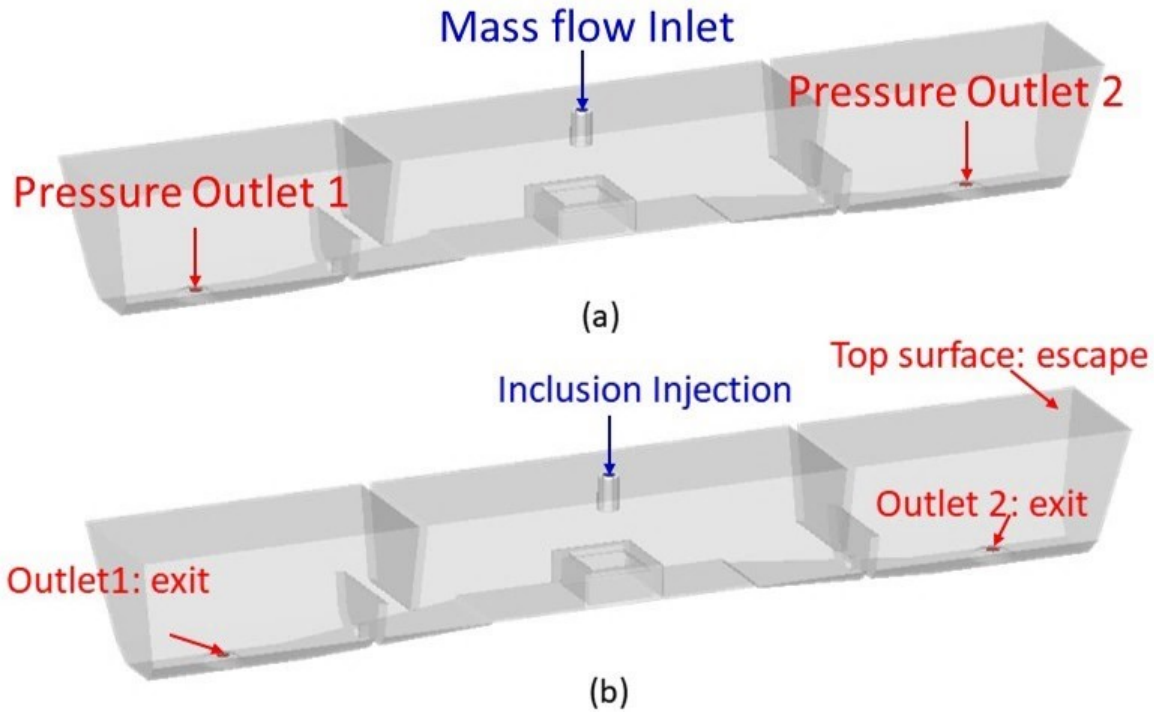


Figure 8.3 Boundary Conditions for (a) fluid flow and (b) particle transport

The heat flux bounday conditions are shown in Table 8.1 [60].

Table 8.1 Thermal Boundary Conditions

Parameters	Symbols	Value
Inlet temperature (K)	q_{in}	1806 / 1836
Outlet backflow temperature (K)	q_{back}	1796 K
Top surface (kW/m^2)	q_{top}	15
Bottom surface (kW/m^2)	q_{bottom}	1.4
Longitudinal wall (kW/m^2)	q_{long}	3.2
Transverse wall (kW/m^2)	q_{tran}	3.8
Weir and dam wall (kW/m^2)	q_{wd}	1.75
Impact box wall (kW/m^2)	q_{im}	0

8.5 Casting Conditions

Four cases are simulated, the first three cases (case S / S10 / S30) are symmetrical tundish with iso-thermal, 10 K superheat, and 30 K superheat conditions. The fourth case is an impact box misalignment case (case M). The casting conditions for all the cases are listed in Table 8.2.

Table 8.2 Caster Dimensions and Process Conditions

Parameters	Symbols	Case S / S10 / S30	Case M
Tundish liquid height (mm)	h_t	1200	
Steel flow rate (metric ton/min)	m	5.8	
Casting speed (m/s)	v_c	0.01	
Molten steel density (kg/m^3)	ρ	7000 or Boussinesq approximation	
Molten steel dynamic viscosity (kg/m·s)	μ_0	0.0063	
Liquidus Temperature (K)	$T_{steel,liq}$	1796	
Inlet superheat (K)	$\triangle T_{sup,in}$	iso-thermal / 10K / 30K	iso-thermal
Steel heat capacity (J/kg·K)	c_p	680[13]	
Steel thermal conductivity (W/m·K)	k_0	26[13]	
Steel thermal expansion coefficient (K^{-1})	β	1.0x10$^-$4[13]	
Inclusion density (kg/m^3)	ρ_p	5000	
Inclusion heat capacity (J/kg·K)	$c_{p,p}$	718	

8.6 Steady-State Symmetrical Flow Results

Figure 8.4 (see page 126) shows the steady-state flow pattern inside the tundish. The flow pattern is symmetrical about its center plane as seen in Figure 8.4 (a). The jet hits the bottom of the impact box and spreads radially. The flow is deflected upside the walls of impact box and up towards the top surface of the tundish middle zone, which may remove inclusions into the top slag layer, as shown in Figure 8.4 (b) and (d). Figure 8.4 (c) shows the flow near the weir-dam region. The weir-dam design forces the steel to flow up to the top slag surface in the outer tundish zones, which facilitates the removal of inclusions.

Figure 8.5 (see page 126) shows the pressure distribution on the tundish outlets. Except in the boundary cells, the outlet pressure is 78970.5 Pa, which exactly matches the ferrostatic pressure above it. The mass flow rates at the two outlets are almost equal, and the mass balance error in this model is only ∼0.000016%.

8.7 Superheat Simulation Results

Figure 8.6 (see page 127) shows how the superheat affects the flow pattern. Apparently, superheat makes the two upper recirculation zones bigger due to thermal buoyancy, and makes the jet below the weir flow upwards. The flow patterns with 10 K and 30 K superheat are almost the same.

Figure 8.7 (see page 127) shows the temperature distribution with 10 K and 30 K superheat. The temperature differences are both ∼6 K across the domain for 10 K and 30 K superheat cases.

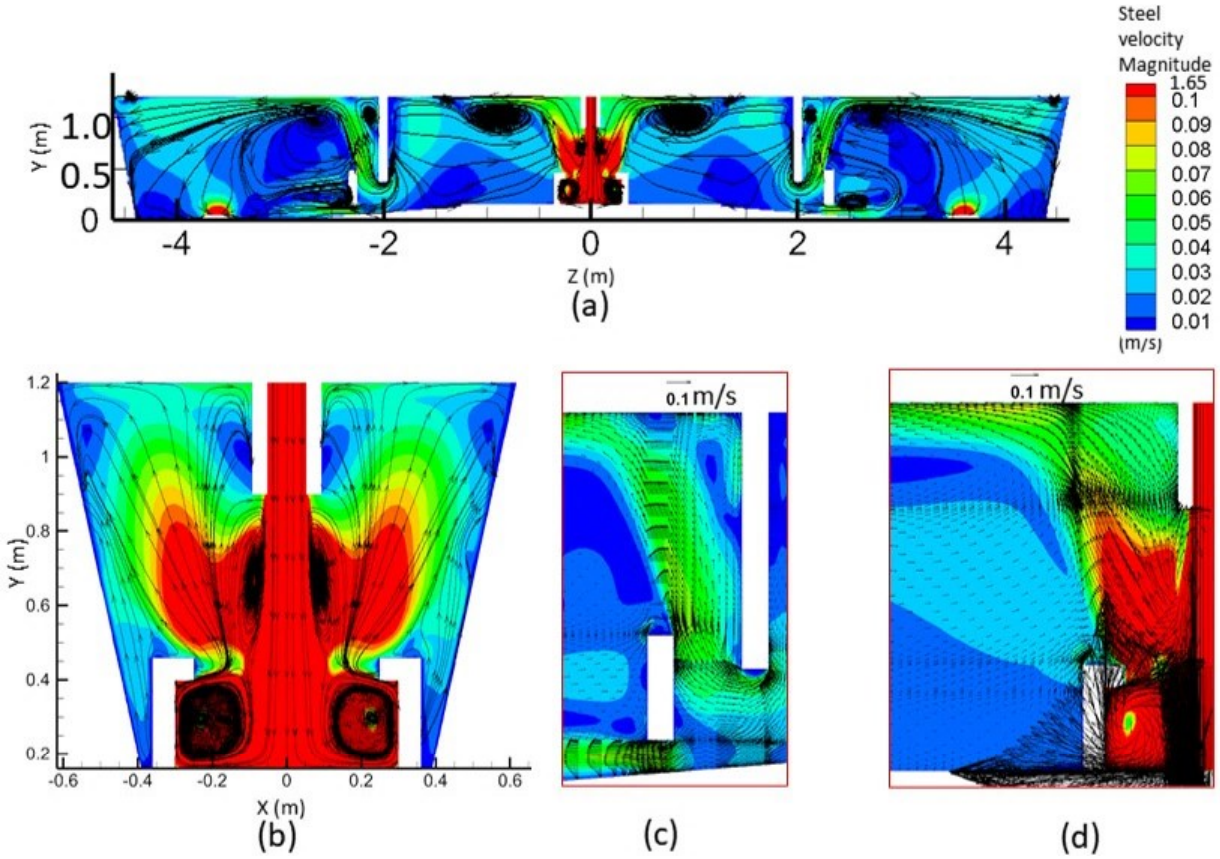


Figure 8.4 Fluid flow in the tundish: (a) Longitudinal front view, (b) Transverse side view, (c) Near the wei-dam region, and (d) Near the impact box region

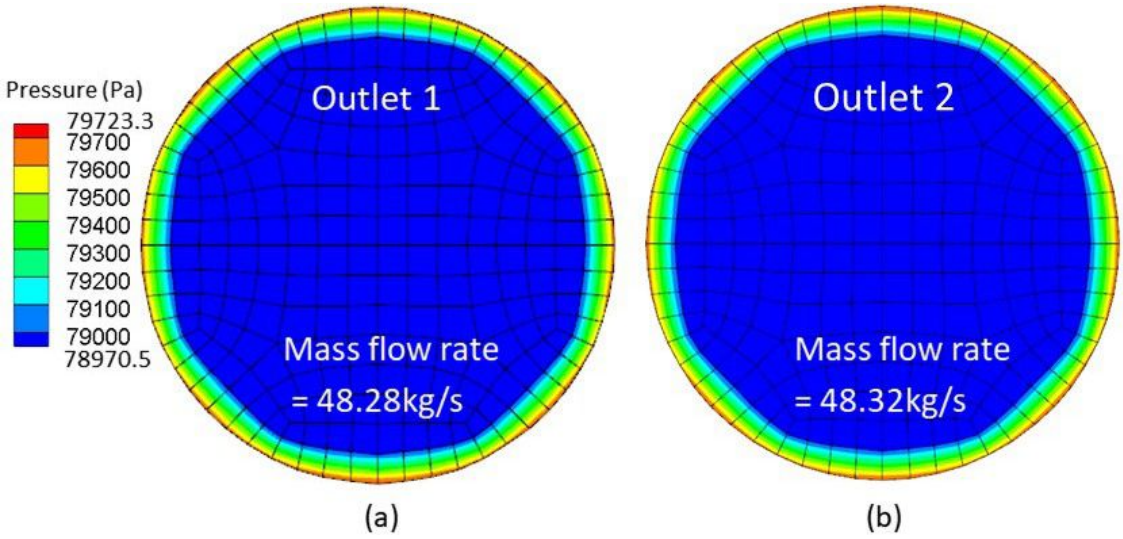


Figure 8.5 The pressure distribution and mass flow rate on tundish outlets

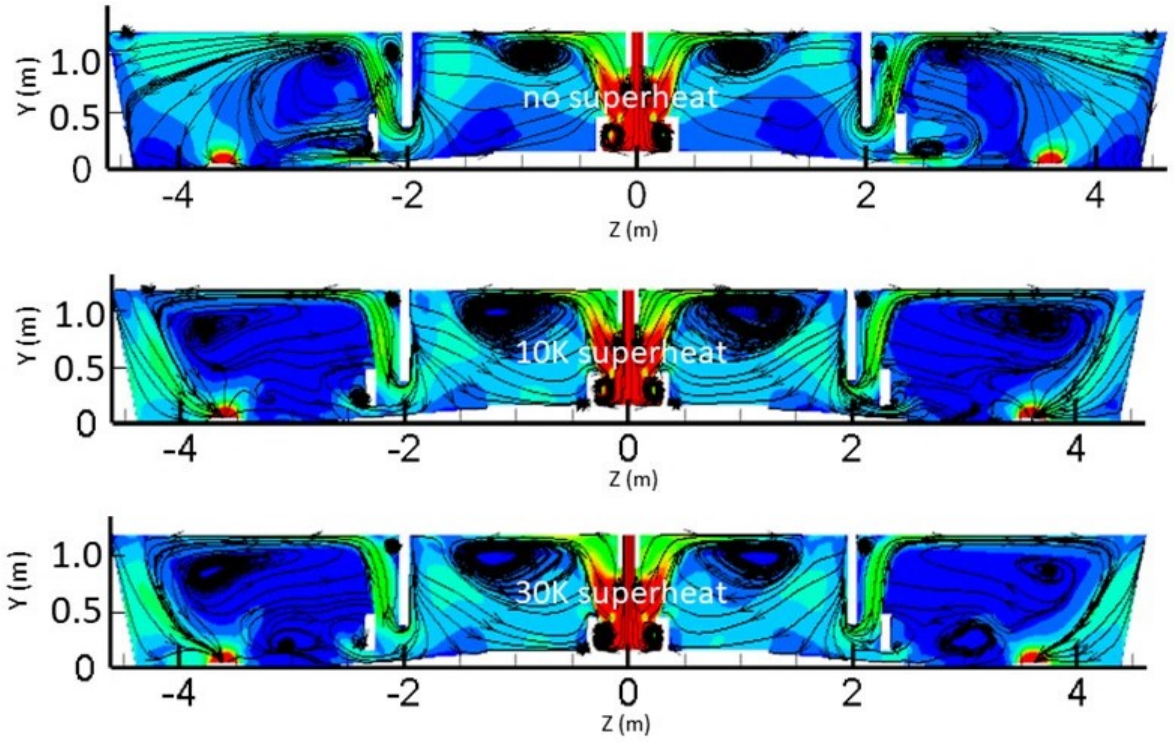


Figure 8.6 Tundish flow pattern with iso-thermal, 10 K superheat, and 30 K superheat conditions

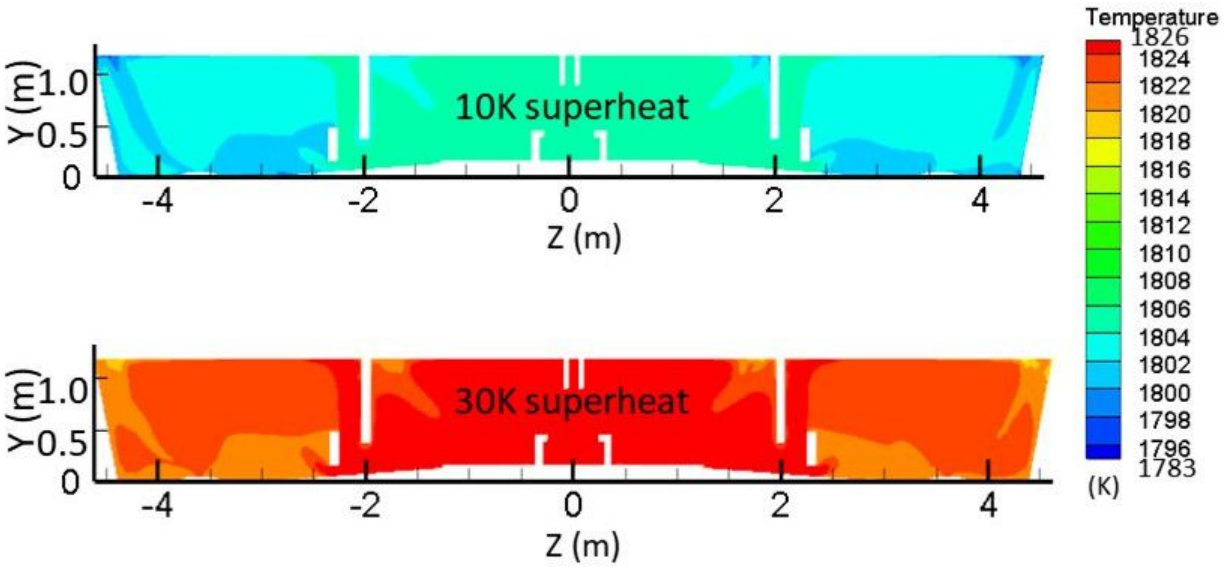


Figure 8.7 Temperature distribution on the center plane of tundish

Figure 8.8 shows the heat loss on each wall of the tundish. The heat loss on all surfaces are identical for 10 K and 30 K superheat cases, due to the same heat flux and surface area.

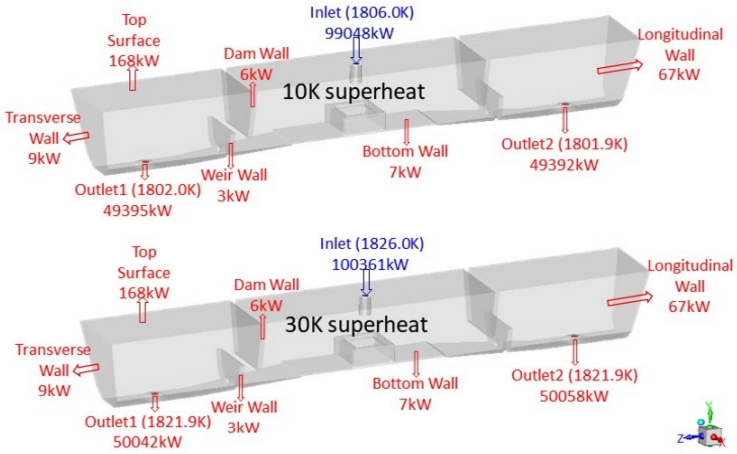


Figure 8.8 Super distribution on the tundish inlet, outlet, and walls

Figure 8.9 shows the exit fraction of different inclusions in the 10K superheat case. The exit fraction of 10 μm inclusions increases from ∼40% in Chapter 7 (standard k-ϵ model) to ∼70% here, possibly due to the finer mesh of the boundary layer near the top surface with SST k-ω model in this chapter, which makes it more difficult for the inclusions to be kicked towards the top surface by the random walk model and be removed. The exit fraction of 100 μm is ∼13%, which is close to that in chapter 7 (12.4%), meaning that the removal of large inclusions are less susceptible to turbulence model and flow pattern.

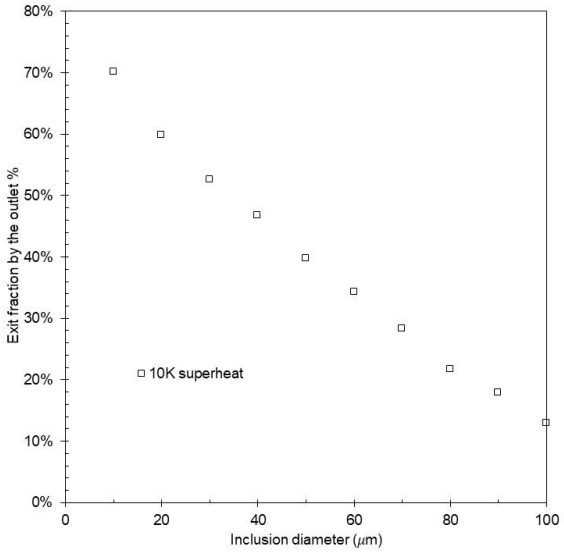


Figure 8.9 Exit fraction of different inclusions in the 10 K superheat case

8.8 Nozzle Misalignment Flow Results

Figure 8.10 compared the flow pattern of case M and case S. In Figure 8.10 (a), the insertion nozzle is moved towards the left for 100 mm. Misalignment makes the flow asymmetrical: the flow from nozzle enters the left side of the impact box and exited upwards from the right side. Higher flow velocity towards surface is seen in the region where insertion nozzle moves away from, indicating more inclusion removal there. Misalignment has no effect on mass flow rate at outlets, because the flow velocity at outlet is governed by the gravitational potential ρgh above. The mass flow rates at the left and right outlet in case M are 48.26 kg/s and 48.34 kg/s.

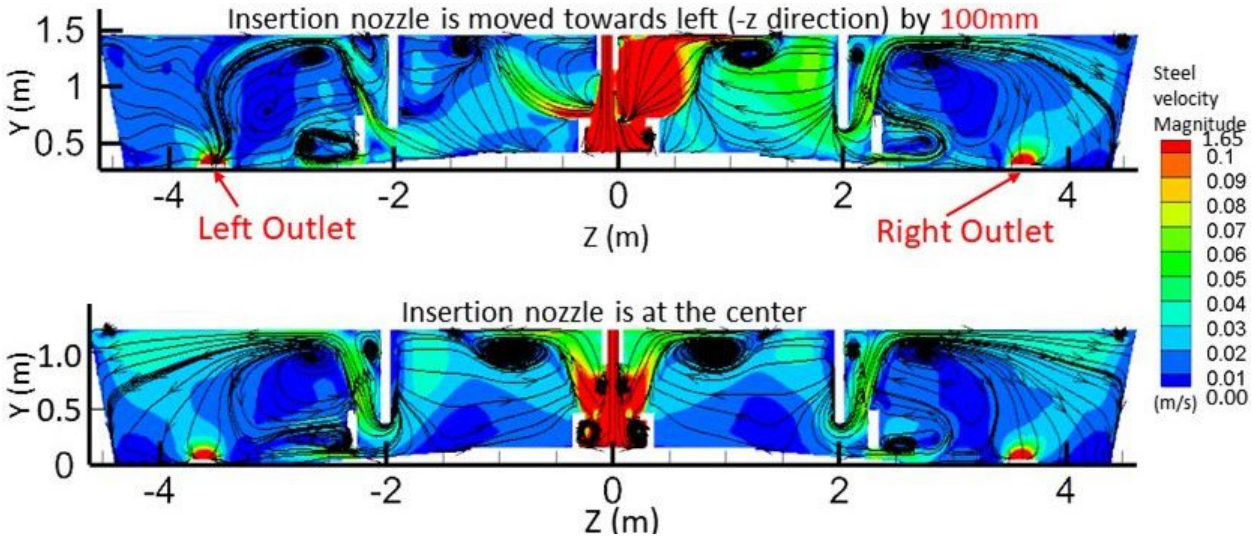


Figure 8.10 Flow pattern of case M and case S

Figure 8.11 shows a 3-d view of the streamline inside the tundish. The jet from the insertion nozzle hits the impact box and gets separated into two streams. The two streams come back together and move rightwards (+z). This indicates that more inclusions will enter the region where the nozzle moves away from. In this case, it is the right half of the tundish (+z half).

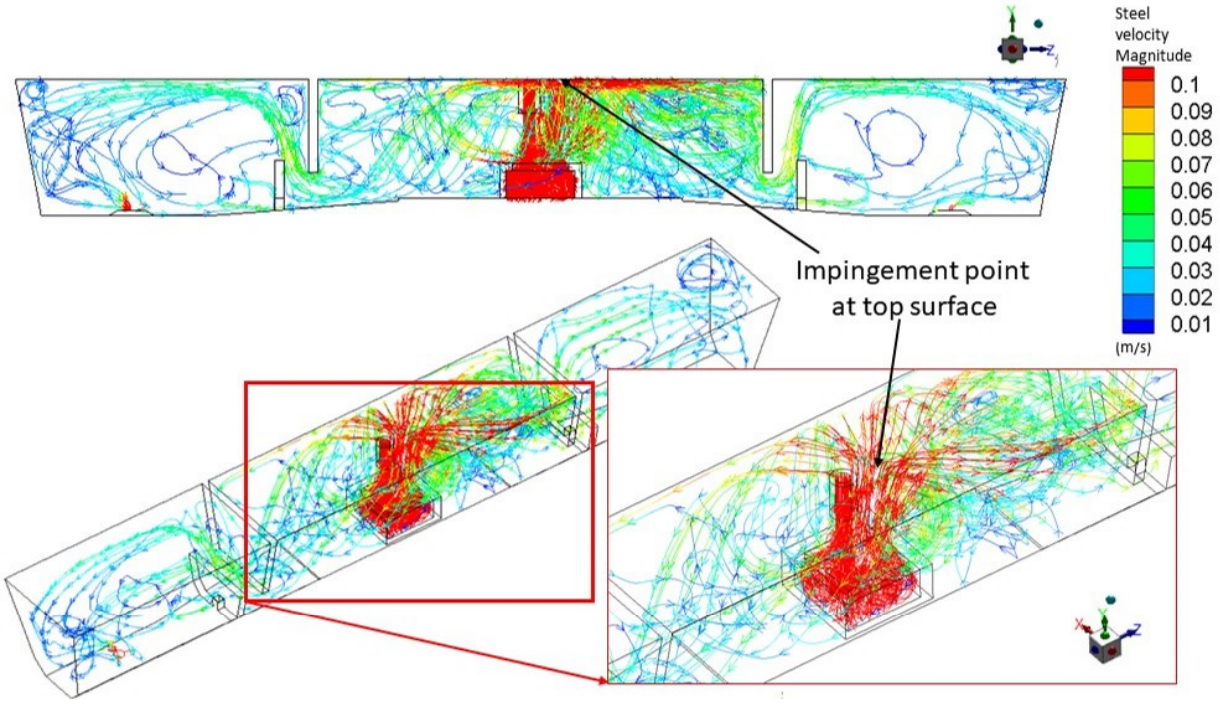


Figure 8.11 3-D view of the streamline in case M

8.9 Inclusion Transport and Removal Results

After getting the steady state flow pattern in case S and M, a burst of 10,000 inclusions are injected at the nozzle inlet for each of the 10 inclusion diameters: 10 μm, 20 μm, 30 μm, 40 μm, 50 μm, 60 μm, 70 μm, 80 μm, 90 μm, 100 μm.

Figure 8.12 shows the inclusions fate in case M. At 400 s, ~32% more inclusions exit the right outlet (7.2%) than the left outlet (4.9%). At 2000s, almost the same number of inclusions exit both outlets with only ~4% more on the left outlet (19.4%) than the right outlet (17.2%), because of higher steel velocity on the right side which leads to less inclusion residence time. More inclusions are removed at the right top surface slag layer, as a results of the impingement from the deflected flow by the bottom impact box.

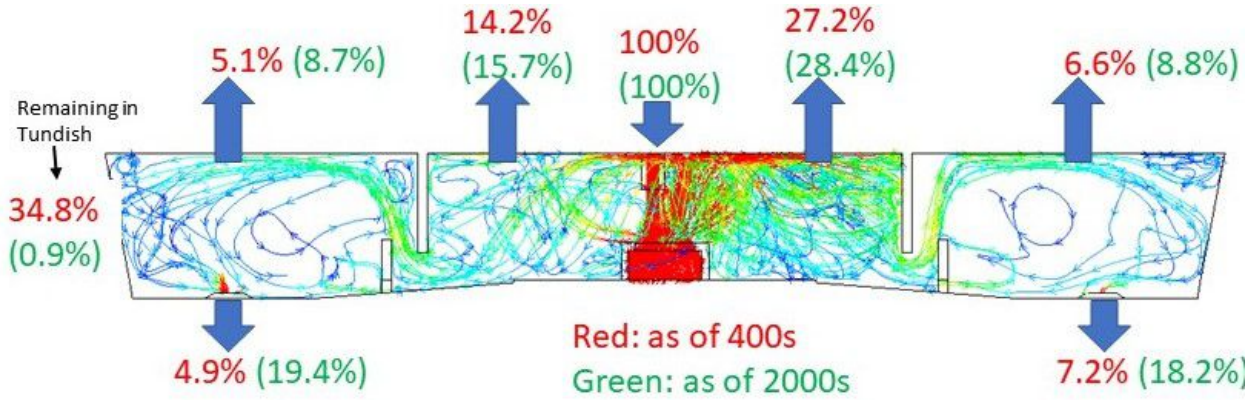


Figure 8.12 Inclusion exit and removal fraction at different time

Figure 8.13 shows the exit fraction of inclusions at the left and right outlets of the tundish for all sizes. The exit fraction is defined as the ratio between the number of inclusions exiting the tundish outlet and the number of inclusions injected. The overall exit fraction reaches steady state at 1600 s. when t < 900 s, more inclusions exit the right outlet due to shorter residence time, for t > 900s, more inclusions exit the left outlet, at steady state, the left outlet has more inclusions exiting than the right.

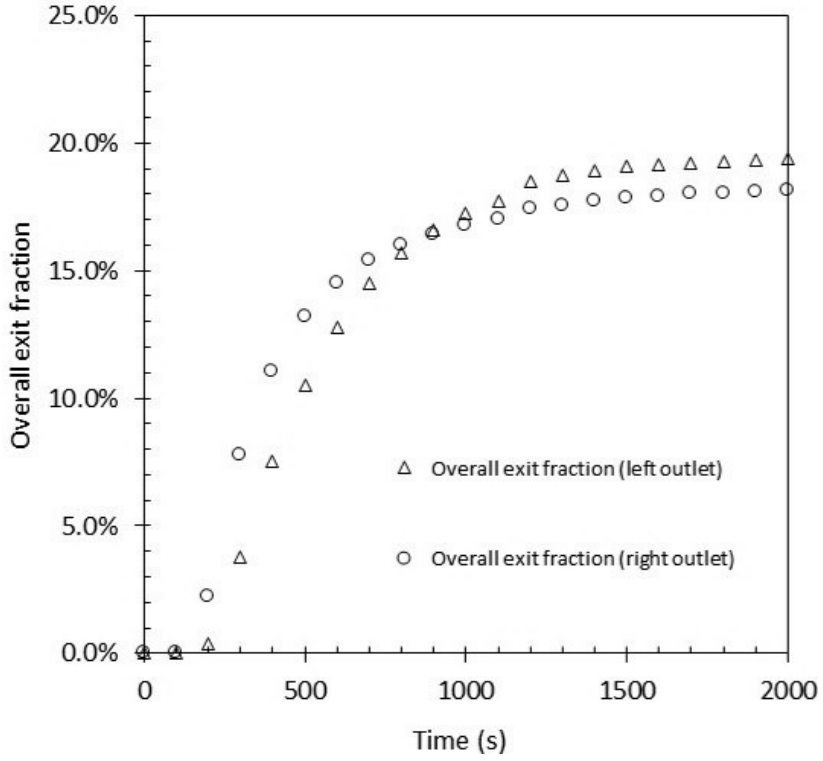


Figure 8.13 Overall exit fraction of case M

Figure 8.14 compared the exit fraction at one outlet of different inclusions between case S and case M. For case M, fewer small ($< 70\ \mu m$) inclusions exit the right outlet due to more removal into the right top surface. 10 - 30% more large ($> 70\ \mu m$) inclusions exit the right outlet due to shorter residence time to float up. Comparing case M with case S, fewer small inclusions ($<= 80\ \mu m$) exit both outlets, and more large inclusions ($> 80\ \mu m$) exit the right outlet.

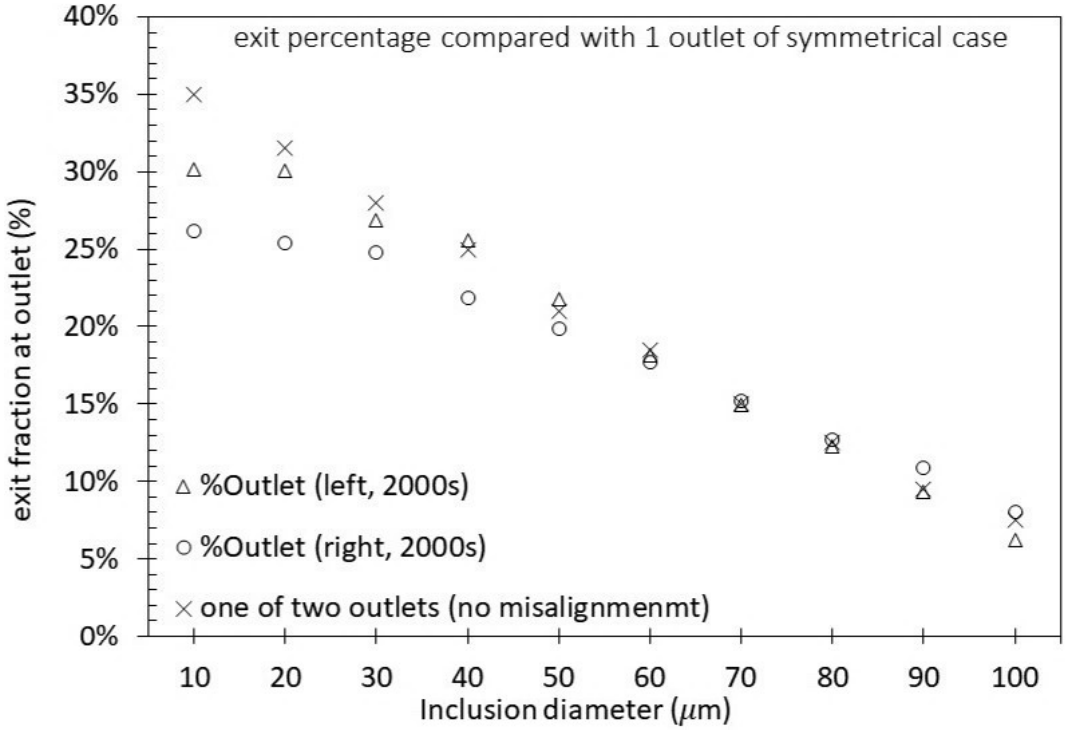


Figure 8.14 Comparison of the exit fraction between case S and case M for different inclusions

8.10 Conclusions

Effect of turbulence model:

- The exit fraction of 10 μm inclusions increases from ~40% in Chapter 7 (standard k-ϵ model with coarser mesh) to ~70% here, possibly due to both the change in flow pattern and the finer mesh of the boundary layer near the top surface with the SST k-ω model in this chapter, which makes it more difficult for the inclusions to be kicked towards the top surface by the random walk model and be removed.

- The exit fraction of 100 μm is ~13%, which is close to that in chapter 7 (12.4%), meaning that the removal of large inclusions are less susceptible to turbulence model and flow pattern.

Effect of superheat:

- The flow patterns for 10K and 30K cases are almost identical.
- For both cases, flow from the impact box and flow between the weir and dam are deflected upwards, thus facilitating the removal of inclusions into slag layer.
- The flow exiting the hole in the dam flows upwards due to thermal buoyancy.
- The temperature difference across the domain is 6K for both cases.

Effect of misalignment:

- The mass flow rate is still the same at two outlets, due to the gravitational potential ρgh above it.
- Misalignment makes the number of large inclusions ($> 70\ \mu m$) exiting the outlet (further from impact box) increase by 10-30% compared with the other side.
- compared with symmetry case, misalignment leads to less small inclusions ($<= 80\ \mu m$) exiting both outlets and more large inclusions ($> 80\ \mu m$) exiting the right outlet.

CHAPTER 9

SUPPLEMENTAL WORK ON THE ENTRAPMENT / ENGULFMENT USER DEFINED FUNCTION

9.1 Introduction

The Entrapment / Engulfment UDF used in Chapter 4 is shown in Appendix A.5. In this UDF, the near-wall cell-center fluid velocity is used as the particle-center fluid velocity, i.e. u_s in Equation 4.37 and 4.38, for the drag and lift forces calculation. This velocity u_s is very close to the particle velocity u_p, except that the particle has a random turbulent velocity (u_i' from Equation 4.28) from the RWM, on top of u_s. This means that, this method makes the drag and lift forces depend on local turbulence only. Equation 4.37 and 4.38 become:

$$F_D = \frac{1}{8}\pi\rho d^2 C_{D3}|u_i'|(u_i') \tag{9.1}$$

$$F_L = -\frac{-9}{\pi}\mu_s R_p^2(u_i')sgn(G)\left[\frac{|G|}{v_s}\right]^{\frac{1}{2}} J^u \tag{9.2}$$

Although this UDF gives a good validation with the UT maps in Chapter 4, improvement is needed to calculate the particle-center fluid velocity, which should be much slower than the cell-center fluid velocity, especially for small particles. Then the instant force balance of a particle touching the solidification front can be reasonably calculated.

This chapter proposed a new method to calculate the particle-center fluid velocity by using the relation between u_s^+ and y^+ from the standard wall function [67, 95], which assumes:

if $y^+ < 11.25$, linear relation for the linear or viscous sub layer (the fluid layer in contact with a wall):

$$u_s^+ = y^+ \tag{9.3}$$

if $y^+ > 11.25$, log law for the turbulent region close to a wall:

$$u_s^+ = \frac{1}{\kappa_s} ln(E_s y^+) \tag{9.4}$$

where κ_s is von Kármán constant, 0.4187, and E_s is an empirical constant, 9.793.

The standard wall function has been widely used in industry to minimize computational time, because it enables the use of coarse cells near the boundary and estimates the near-wall velocity by the linear and log law relations.

When a particle touches the solidification front, the y^+ value is firstly calculated based on the distance between the solidification front and the particle center (which is estimated as the particle radius R_p). Then, u_s^+ at the particle center can be computed from y^+, and the particle-center fluid velocity magnitude u_{sm} can be calculated. The u_{sm} is decomposed in x, y, and z directions based on the cell center velocities, and then the drag and lift forces are calculated.

9.2 Model Equations and Procedures

u_s^+ and y^+ are defined as follow [123]:

$$u_s^+ = \frac{u_{sm}}{u_\tau} \tag{9.5}$$

$$y^+ = \frac{u_\tau d_p \rho_s}{2\mu_s} \tag{9.6}$$

where u_{sm} is the steel velocity magnitude at the particle center (m/s). u_τ is the friction velocity and is defined as:

$$u_\tau = \sqrt{\tau_w/\rho_s} \tag{9.7}$$

where τ_w is the wall shear stress.

if $y^+ < 11.25$, substitute Equations 9.5 and 9.6 into Equations 9.3:

$$u_{sm} = \frac{\tau_w d_p}{2\mu_s} \tag{9.8}$$

if $y^+ < 11.25$, substitute Equations 9.5 and 9.6 into Equations 9.4:

$$u_{sm} = \frac{ln(\frac{E u_\tau d_p \rho_s}{2\mu_s}) u_\tau}{\kappa_s} \tag{9.9}$$

Then the x, y, and z fluid velocity at cell center can be calculated by:

$$u_{spx} = \frac{u_{scx}}{u_{scm}} u_{sm} \tag{9.10}$$

$$u_{spy} = \frac{u_{scy}}{u_{scm}} u_{sm} \tag{9.11}$$

$$u_{spz} = \frac{u_{scz}}{u_{scm}} u_{sm} \tag{9.12}$$

where u_{spx}, u_{spx}, and u_{spx} are the x, y, and z fluid velocity at particle center (m/s). u_{scx}, u_{scx}, and u_{scx} are the x, y, and z steel velocity at the near-wall cell center (m/s). u_{scm} is the near-wall cell-center velocity magnitude.

With u_{spx}, u_{spx}, and u_{spx}, the velocity difference between the particle and the fluid at the particle center is calculated, and the drag and lift forces are computed.

9.3 Computational Details

30,000 bubbles of each different size are injected and tracked by the DPM model into the steady-state Eulerian-Eulerian flow from case A10 in Chapter 4. Their capture is determined by hook capture and both the old and new entrapment / engulfment UDFs.

The old and new entrapment / engulfment UDFs are listed in appendix A.5 and A.6.

9.4 Results - Particle Capture

Figure 9.1 (see page 137) shows the capture locations with the hook capture and old and new entrapment / engulfment UDFs on WFIR. For bubbles < 0.3 mm, the capture locations are very similar from old and new UDFs, because almost all capture are caused by entrapment. For bubbles $>= 0.3$ mm, there's less capture near the jet region. The new UDF underpredicts 0.6 mm bubble captured inside the capture-band region shown in Chapter 4, possibly due to the neglect of transient turbulence.

Figure 9.2 (see page 137) shows the capture locations with the hook capture and old and new entrapment / engulfment UDFs on WFOR. Same trend is seen. It is more prominent that there's less capture of bubbles $>= 0.3$ mm with the new UDF, as a results of higher velocity near the jet washing bubbles away.

Figure 9.3 (see page 138) shows the y^+ value at the center of 0.6 mm bubbles on WFIR and WFOR. The maximum y^+ is $\sim 9 < 11.25$, meaning that all the bubbles $<= 0.6$ mm are inside the sub viscous layer when touching the solidifying shell.

Figure 9.4 (see page 138) puts the bubble capture locations and bubble-center velocity distribution side by side. There's less capture on WFOR due to larger near-jet velocity washing the bubbles away.

Figure 9.5 (see page 139) and Figure 9.6 show the capture locations of 0.4 mm, 0.5 mm, and 0.6 mm bubbles. For 0.4 mm and 0.5 mm bubbles, they are captured all by entrapment, forming a sudden capture line.

Figure 9.7 (see page 140) shows that capture fraction of different bubble sizes with old and new UDFs. The new version gives lower capture fraction for bubbles $>= 0.3$ mm. For bubbles < 0.3 mm, the capture fraction is similar.

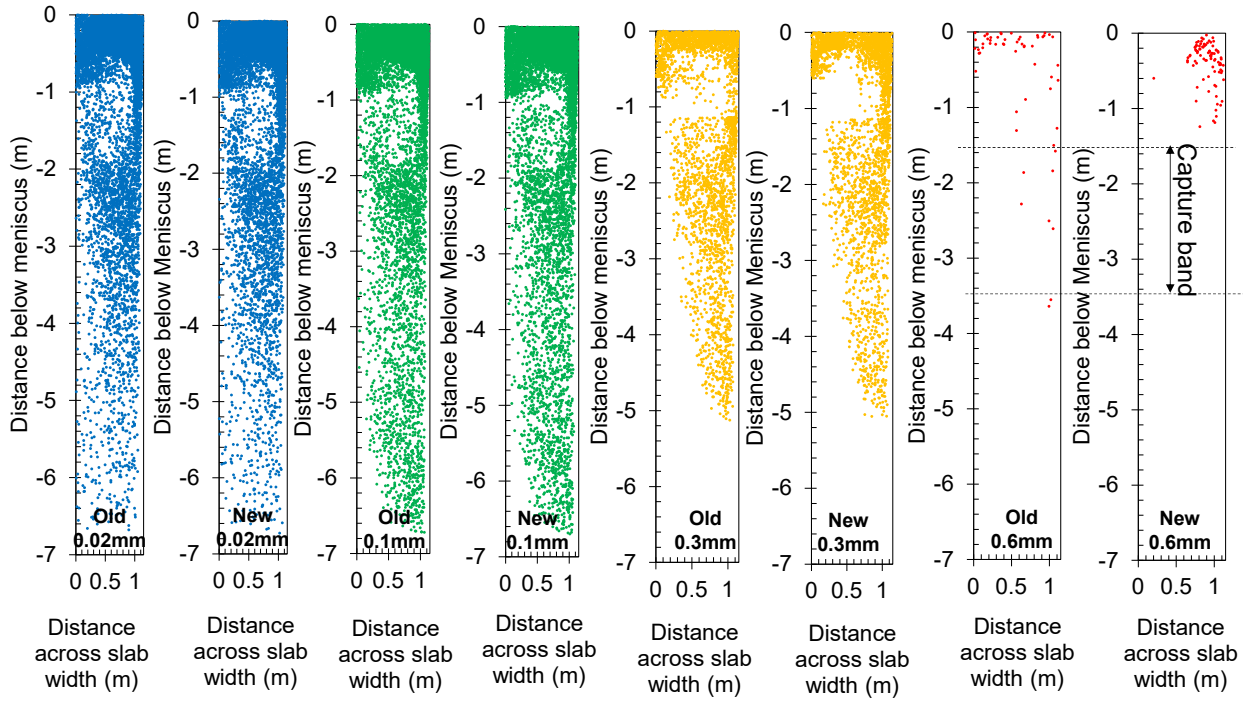


Figure 9.1 Bubble capture locations on WFIR

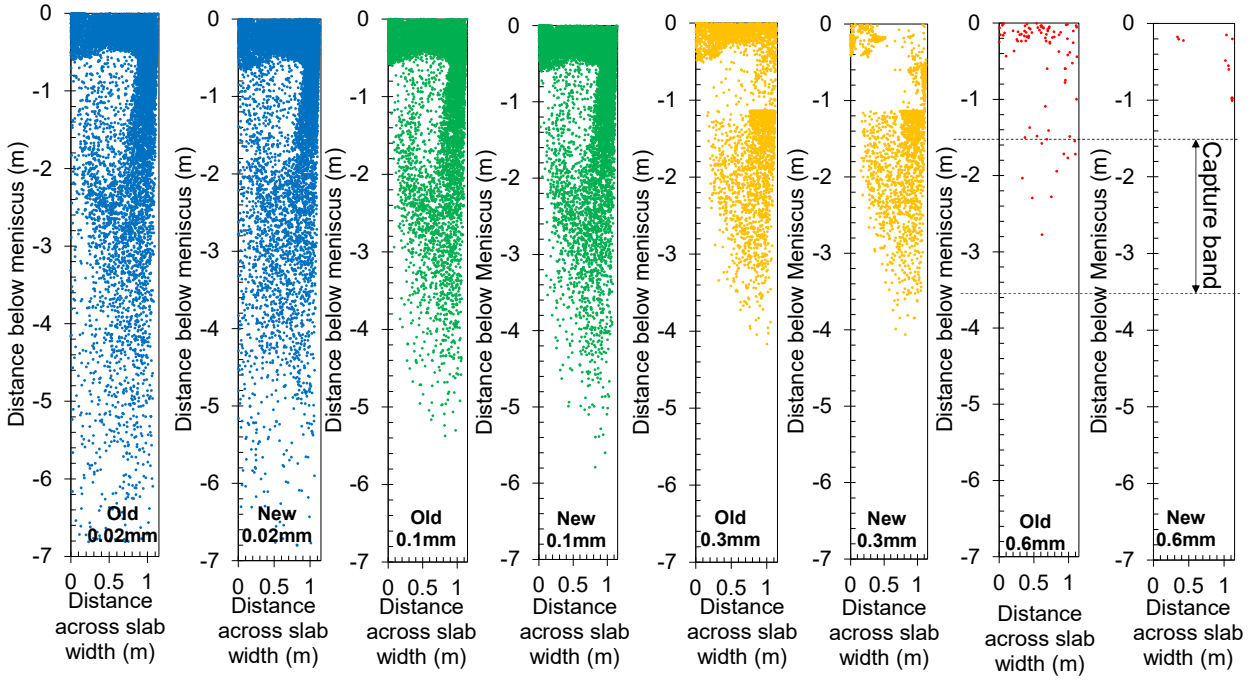


Figure 9.2 Bubble capture locations on WFOR

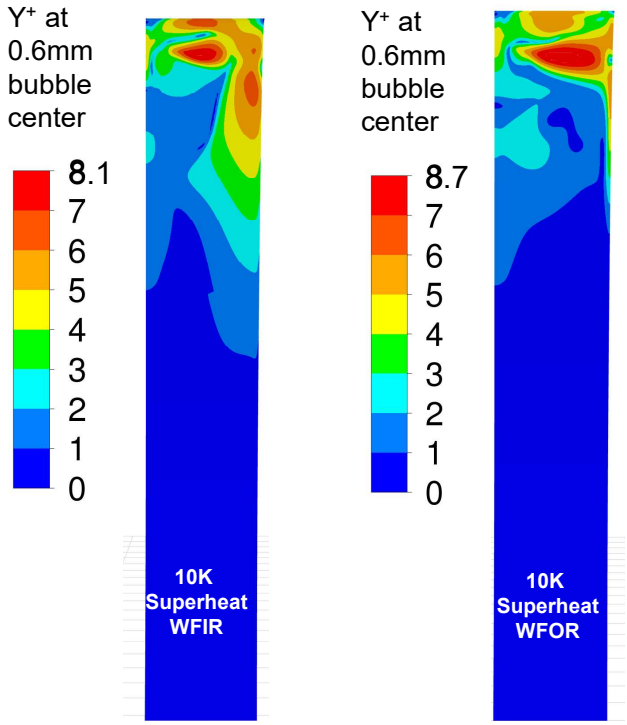


Figure 9.3 y^+ value at 0.6 mm bubble center

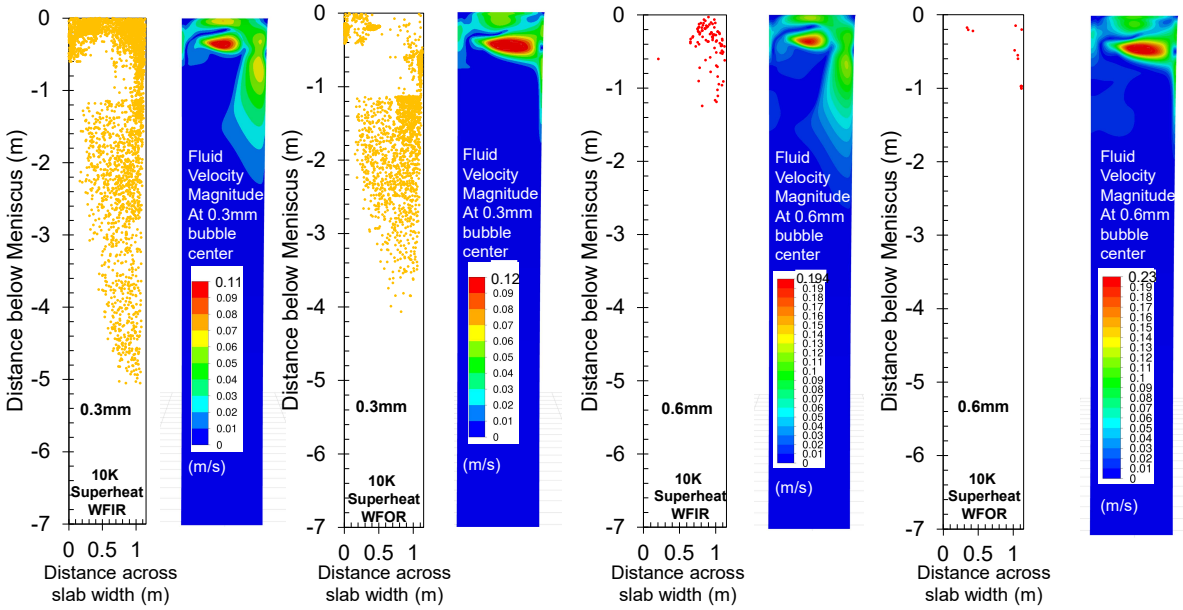


Figure 9.4 Bubble capture locations and bubble-center velocity on WFIR

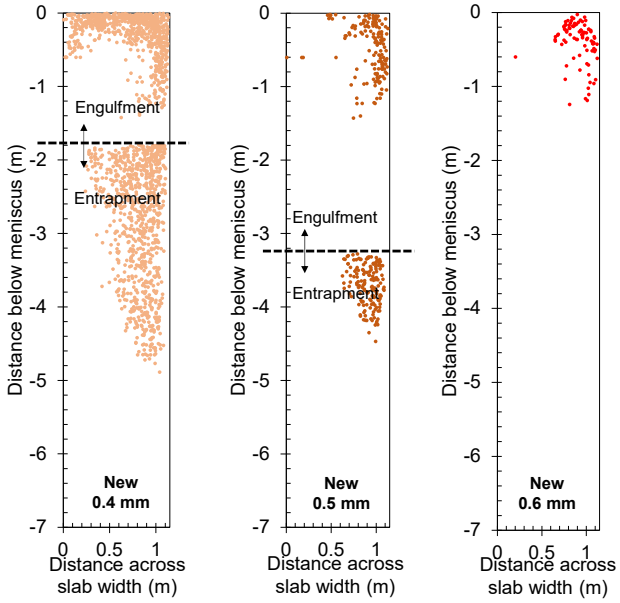


Figure 9.5 Capture locations on WFIR for 0.4 mm, 0.5 mm, and 0.6 mm bubbles

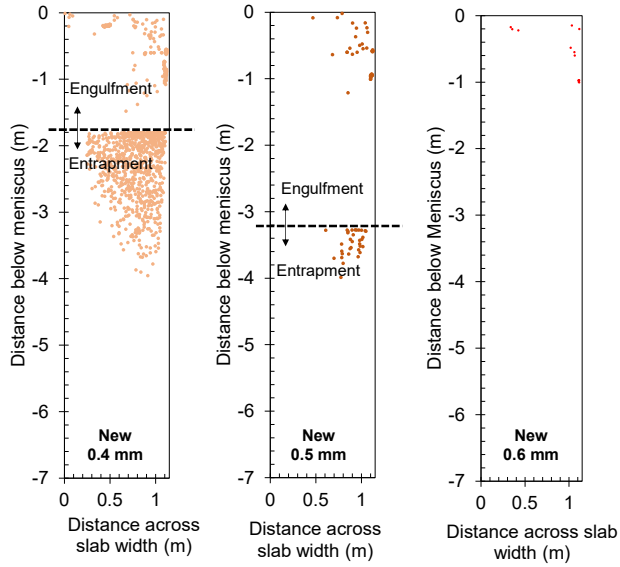


Figure 9.6 Capture locations on WFOR for 0.4 mm, 0.5 mm, and 0.6 mm bubbles

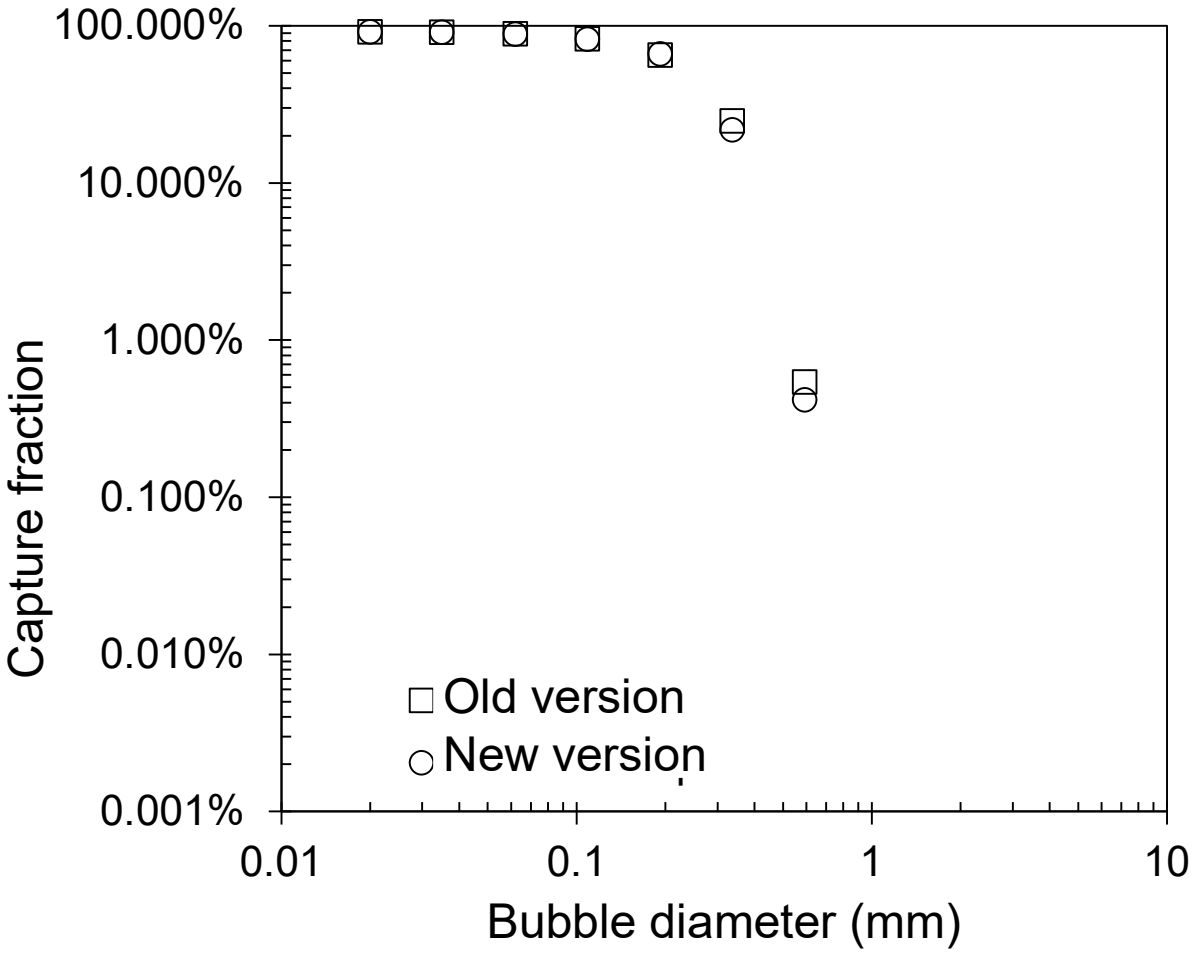


Figure 9.7 Capture locations on WFOR for 0.4 mm, 0.5 mm, and 0.6 mm bubbles

9.5 Conclusions

A supplemental new method is developed to calculate the particle velocity when it touches the wall and the fluid velocity at particle center, which is used in the calculation of engulfment capture mechanism

- The new method has similar capture fraction and locations for bubbles < 0.3 mm
- The new method has less capture for larger bubbles ($>= 0.3$ mm)
- The new method has less capture near the high-speed jet region, where bubbles got washed away
- However, this new method predicts zero 0.6 mm bubbles captured in the plant-measured capture-band region, which is possibly due to the neglect of transient turbulence, which enables near wall velocity variation and bubble capture

CHAPTER 10

CONCLUSIONS AND FUTURE WORK

In this work, a new model system has been developed and applied to investigate the flow pattern, superheat dissipation, particle transport, and capture in a continuous-casting tundish, nozzle, and strand for steel slabs.

10.1 Modeling System

A new computationally-efficient methodology has been developed to calculate both the single-phase and multiphase flow, superheat dissipation, particle transport, and capture in a continuous-casting tundish, nozzle, and strand.

The main new model system has been developed for the nozzle and strand: First, the mean bubble size needed by the multiphase flow model is predicted using a model of pressure distribution and flow rate within a slide-gate flow delivery system (PFSG) model to find the local pressure, and a previously-developed two-stage analytical bubble size model for gas injection into downward flowing metal. With the predicted mean bubble size and a spread parameter from the curve fitting of a water model experiment, the bubble size distribution is obtained by applying these two parameters into a Rosin-Rammler distribution equation. A heat transfer and solidification model (CON1D) is used to estimate the shell thickness profile down the strand, which is characterized with a solidification constant, and used to build the strand liquid pool geometry, implement the mass/momentum/energy sinks due to the steel solidification, and predict the PDAS size evolution. The mass, momentum, and energy sink terms in the UDFs are modified for the curved part of the strand by the author, based on the empirical shell thickness equation. The mass balance error is less than 0.01% if good convergence is achieved.

Then, the multiphase flow in the nozzle and liquid pool inside the strand is computed with an E–E model (Ansys Fluent), including the effects of thermal buoyancy with a fully-coupled heat transfer model of superheat transport, based on solving the scalar energy convection-diffusion equation. A simple energy balance model of the top slag layer was developed to calculate the heat flux lost from the top surface of the molten steel, which successfully predicts the top surface freezing and explains the formation mechanism of the giant center defects as seen in some UT maps. The difficulty of convergence of wide caster (> 2m) is successfully resolved by making the strand mesh aligned with the jet direction. Next, different bubble sizes were injected into the strand flow field and tracked using a Lagrangian DPM model (Ansys Fluent). Previous capture mechanisms (hook/entrapment/engulfment) were implemented together into a new

comprehensive model to determine the capture of argon bubbles. A model of the primary dendrite arm spacing (PDAS) down the caster is developed to evaluate the capture mechanism.

Finally, a new post-processing method is applied to calculate the particle capture fractions and compare them with measurements in the slab samples. With this new model system, the effect of superheat on flow pattern and particle capture is investigated for the first time. The results match plant measurements reasonably.

A multiphase model coupling LES with DPM particles has been developed for simulating the transient asymmetrical flow in the nozzle and strand. However, it would take ∼3 years to get a fully-developed LES flow, especially in the lower strand where accurate particle capture is needed. So, the LES model is not practical in real-world applications for this specific situation, with current computational power.

Next, the new modeling approach has been extended to investigate the effect of mold-electromagnetic stirrer on multiphase turbulent flow, argon gas injection, particle transport, and capture in the strand during steady continuous casting of steel slabs, with a periodic boundary condition for the strand center plane, which simplifies the geometry and helps to achieve better convergence. The electromagnetic force is measured by the plant and interpolated onto each cell as a User-Defined Scalar (UDS). The UDS is then used as a source term in the Navier-Stokes equation. Then, 10,000 3mm-diameter argon bubbles are injected and tracked into this steady-state flow pattern in a one-way-coupled manner. Their fates are determined by the hook/entrapment/engulfment capture mechanisms.

In addition, a coupled model has been developed for turbulent flow and heat transport in the tundish. A single-phase RANS flow is first simulated for the molten steel in the tundish. Then, one-way coupled inclusions are injected at the tundish inlet with DPM model due to their very small total oxygen mass fraction (15 ppm). The inclusions can either escape into the top slag layer or exit the tundish outlet, flowing into the nozzle and strand. With this model, The effects of ladle switching from high to low temperature and insertion nozzle misalignment on tundish flow pattern and inclusion removal are investigated for the first time. The inclusion exit fraction at the tundish outlet versus casting time after ladle switch is quantified.

This thesis has done some creative and innovative work that has never been done before, which can be implemented into future research.

10.2 Conclusions from Simulation Results

The models developed in this work reveal many new and practical insights into continuous casting.

For the nozzle and strand flow, the flow pattern in a caster with two different SEN designs is compared in Chapter 3. The results show that the SEN 1 has an upward flow near the nozzle ports, which tends to

transport more large bubbles into the top slag layer and leads to better steel quality. However, the validation with the nail-board dipping tests is not good.

The same model system is used for SEN 1 again in Chapter 4, where discrete argon bubbles are injected and tracked by the DPM model into the steady-state E-E flow. 10 K and 30 K superheat are used. Good agreement is seen with plant measurements of both nail-board dipping tests and UT maps. And the results suggest:

- Superheat has negligible effect on the flow in the upper strand region where inertia is dominant over thermal buoyancy.
- Superheat has a significant effect in the lower strand, especially > 3 m below the meniscus where thermal buoyancy dominates the flow and creates multiple complicated recirculation regions.
- The flow pattern shows two locations for large bubbles to be engulfed: (1) near the stagnation point on the IR wall, where the bubble tends to stay stationary, and (2) in the IR- and OR- side downwards flow region, where the downwards velocity balances the bubbles' terminal velocities. These two places can form two capture bands, which agrees to plant measurements.
- The lower (10 K) superheat case has a deeper IR-side stagnation point and higher OR-side downwards flow due to less thermal buoyancy which allows steel to flow deeper and faster, indicating deeper capture.
- The lower (10 K) superheat case leads to a possible frozen-steel "island" near the top surface due to lower jet temperature. This "island" may move down, capturing slag, bubbles and inclusions, forming a giant defect in the final product.
- Higher (30 K) superheat leads to a more uniform downwards velocity and a shallower stagnation point on the IR side, due to the lower third recirculation zone pushing flow upwards.
- With lower (10 K) superheat, more 0.6 mm bubbles are captured on the OR side in the capture-band region than the IR side, due to the higher downwards velocity transporting the bubbles deeper and balancing out their terminal velocities (0.213 m/s).
- The lower (10 K) superheat case leads to more captured bubbles by meniscus hook due to deeper hook depth.
- The higher (30 K) superheat case leads to shallower capture of all bubble sizes as a result of stronger thermal buoyancy, indicating less internal defects.

- The higher (30 K) superheat case leads to relatively more 0.6 mm bubbles captured on IR side than OR side in the capture-band region, due to a more uniform downwards flow.
- Capture bands are seen at 1/4 across the thickness on both the IR and OR side.
- For small bubbles ($d_p < 0.5$ mm), clear capture bands are seen on both IR and OR side near the transition line between the straight and curved part of strand from the end view.
- Small bubbles ($d_p < 0.5$ mm) tend to escape into the slag layer uniformly over its surface, while large bubbles ($d_p > 0.5$ mm) escape near the SEN due to their larger buoyancy bringing them up immediately after exiting the nozzle port.
- The escape fraction into the top slag layer increases with increase of bubble sizes due to large buoyancy force (∼9% to ∼100% for 0.02 mm and 0.6 mm bubbles respectively).
- Superheat has little effect on the capture of bubbles smaller than 0.5 mm.
- Increasing superheat decreases the capture of large (0.6 mm) bubbles by 20% from 0.06% for 10 K superheat to 0.05%for 30 K superheat due to the complicated recirculation regions hindering the penetration of large bubbles deep into the caster; All bubbles > 1mm escaped into the top slag layer.
- For $d_p < 0.3$ mm bubbles, entrapment dominates over engulfment, because the bubbles are smaller than PDAS; For $d_p > 0.3$ mm bubbles, engulfment becomes more important.
- Inside the hook zone: For small bubbles ($d_i < 0.5$ mm), entrapment and engulfment criterion is > 20× more important than hook capture mechanism; For 0.6 mm bubble in lower (10 K) superheat case, they are of the same importance; All captured 1 mm bubbles are captured by hook capture mechanism in the low (10 K) superheat case.

Even though higher superheat leads to less and shallower capture of large particles, lower superheat is very beneficial for centerline quality due to better macro-segregation, solidification shrinkage, and thermal / solutal gradient. Companies need to choose between cleanliness and centerline quality.

In Chapter 5, the LES model is used to simulate the transient asymmetrical fluid flow, initialized with the single-phase RANS flow pattern. The results show that at 30 s, the solution starts getting asymmetrical, with the right lower recirculation zone pushing to the left. From 40 s - 70 s, small swirls in the lower strand region occur, and the lower recirculation zone gets shorter because thermal buoyancy effect dominates over inertia effect.

In Chapter 6, the investigation including the effects of EMS on flow and particle capture reveals:

- Effect of EMS force on flow pattern in nozzle is negligible, due to the small EM force in mold center region and high velocity in nozzle which dominates the flow.
- EMS causes impingement point on NF to go deeper, because the EM force pushes the flow down.
- The counterclockwise EM force causes molten steel mold surface flow to accelerate where EM force adds to flow from NF to nozzle and to decelerate where EM force cancels flow from nozzle to NF.
- With EMS on, fewer bubbles are captured in hook zone (6mm below the strand surface) by advanced capture criterion and hook mechanism respectively, due to high velocity near WFs. But $\sim 30\%$ more bubbles are captured by all mechanisms with EMS (53%) than without EMS (38%), because EM force pushes the jet down.

Chapter 7 investigates the effect of ladle switching from high to low superheat / temperature on the tundish flow pattern and inclusion transport and removal. The particle exiting the tundish outlet changes when the successive ladles have different superheat / temperature. This would naturally change the results, by changing the number of inclusions entering the mold.

Pouring hot/cold steel into hot/cold steel (same temperature) will have not effect on the flow pattern and inclusion removal.

Pouring hot steel into cold steel can slightly facilitate inclusion removal because the hot steel tends to flow upwards (hypothesis).

Pouring cold steel (10K superheat) into hot steel (30K superheat) has negative effect on flow pattern:

- Cold steel tends to flow downwards and brings the three jets down, leading to more inclusion exiting outlet.
- Both velocity and temperature distribution reach quasi-steady state after 2,700s of decreasing inlet temperature, which is about $4\times$ longer than the regular steel residence time (707s) due to jets stay down, forming a stratified flow.

For small inclusions ($\leq 100\mu$m):

- Largest exit fraction occurs for inclusions ($\leq 100\mu$m) injected between 50s – 300s.
- Smaller the inclusion, higher the exit fraction because smaller particles are easier to flow with the jet.
- Initial rise of exit fraction after decreasing inlet temperature by 20K is due to Jet1, 2 and 3 going downwards; The first drop in exit fraction is due to Jet1 going back up; Constant exit fraction is due

to jet 2 and 3 maintaining downwards for a long period of time; finally, the second drop of exit fraction is due to jet 2 and 3 turning back up,

- For 10 μm inclusions, the exit fraction increases $\times$2 times; while for 100 μm inclusions, it increases $\times$3 times.

For larger inclusions (>100μm):

- Jet 1's going down has negligible effect on inclusions (>100μm) because of its high velocity/inertia effect towards top surface and large inclusions' larger buoyancy force.
- Jet 2 and 3 dominate over jet 1 in terms of inclusions (>100μm) exit fraction, which is reflected as the almost constant exit fraction for inclusions injected before 1500s.
- The exit fraction decreases for inclusions injected after 1500s because Jet 2 starts to turn upwards at 1600s.
- Decreasing ladle temperature by 20K can make the exit fraction of 200μm and 300μm inclusions 32 and 80 times larger than steady-state happening when feeding the same-temperature steel or hotter steel).

This effect of switching ladle from high to low temperature will cancel off the good effect of superheat on the removal of large particle as seen in Chapter 4, leading to worse steel quality even than the keeping using lower superheat steel.

Chapter 8 investigates the effect of turbulence model, superheat, and insertion nozzle misalignment on tundish flow pattern and inclusion removal, with the same temperature in successive ladles.

Effect of turbulence model:

- The exit fraction of 10 μm inclusions increases from ~40% in Chapter 7 (standard k-ϵ model with coarser mesh) to ~70% in Chapter8, possibly due to both the change in flow pattern and the finer mesh of the boundary layer near the top surface with the SST k-ω model in Chapter8, which makes it more difficult for the inclusions to be kicked towards the top surface by the random walk model and be removed.
- The exit fraction of 100 μm is ~13% in Chapter 8, which is close to that in Chapter 7 (12.4%), meaning that the removal of large inclusions are less susceptible to turbulence model and flow pattern.

Effect of superheat:

- The flow patterns for 10K and 30K cases are almost identical.
- For both cases, flow from the impact box and flow between the weir and dam are deflected upwards, thus facilitating the removal of inclusions into slag layer.
- The flow exiting the hole in the dam flows upwards due to thermal buoyancy.
- The temperature difference across the domain is 6K for both cases.

Effect of misalignment:

- The mass flow rate is still the same at two outlets, due to the gravitational potential ρgh above it.
- Misalignment makes the number of large inclusions ($> 70\ \mu m$) exiting the outlet (further from impact box) increase by 10-30% compared with the other side.
- compared with symmetry case, misalignment leads to less small inclusions ($<= 80\ \mu m$) exiting both outlets and more large inclusions ($> 80\ \mu m$) exiting the right outlet.

10.3 Future Work

Firstly of all, more validation data from the plant is needed for the M-EMS and tundish simulation, by applying these models to other casters / conditions where multiple sets of plant measurements are available.

For the LES simulation, more time (longer than 6 months) is needed for it to reach quasi-steady state, and then the particle capture fraction can be analyzed.

To better model the argon bubbles injected through the stopper rod with large gas fractions, where slug flow and large gas pockets may form, a EEDPM model developed by H. Yang[117] can be used to predict the mean bubble size and bubble size distribution. This model combines the Eulerian - Eulerian multiphase model with the Discrete Phase Model (DPM) in a two-way coupled way. The bubble expansion, coalescence, and breakup are considered in the DPM model, which can be used to determine the local bubble size in the continuous phase of Eulerian-Eulerian model at each time step.

A supplemental ongoing work of the implementation of the entrapment / engulfment capture criteria UDF is explained in Chapter 9. This UDF sets the particle velocity at the casting speed and interpolates the fluid velocity at the particle center according to the standard wall function. The results show less capture for large bubbles (>= 0.3 mm) near the high-speed jet region, where particles got washed away. However, this new method underpredicts the number of bubbles captured inside the capture band for the example investigated and needs more research and application.

REFERENCES

[1] Brian G. Thomas. Continuous casting. *Encyclopedia of Materials: Science and Technology*, pages 1595–1598, 2001.

[2] Steel statistical yearbook 2017, 2017. URL Lhttps://www.worldsteel.org/en/dam/jcr:3e275c73-6f11-4e7f-a5d8-23d9bc5c508f/Steel+Statistical+Yearbook+2017.pdf.

[3] Brian G. Thomas. Introduction to continuous casting, 2022. URL http://ccc.illinois.edu/introduction/concast.html.

[4] G. C. Duderstadt, R. K. Iyengar, and J. M. Matesa. Tundish nozzle blockage in continuous casting. *The Journal of The Minerals, Metals Materials Society*, **20**(2):89–94, 1968.

[5] Brian G. Thomas. Slab casting phenomena, 2022. URL http://ccc.illinois.edu/introduction/basicphenom.html.

[6] M. Liang, SM. Cho, H. Olia, L. Das, and B.G. Thomas. Modeling of multiphase flow and argon bubble entrapment in continuous slab casting of steel, 2019.

[7] B. G. Thomas. Review on modeling and simulation of continuous casting. *steel research int.*, **89**, 2018.

[8] J. L. Shen, D. F. Chen, X. Xie, L. L. Zhang, Z. H. Dong, M. J. Long, and X. B. Ruan. Influences of sen structures on flow characters, temperature field and shell distribution in 420 mm continuous casting mould. *Ironmaking and Steelmaking*, **40**(4):263–275, 2013.

[9] F. M. Najjar, B. G. Thomas, and D. E. Hershey. Numerical study of steady turbulent flow through bifurcated nozzles in continuous casting. *Metallurgical and Materials Transactions B*, **26**(4):749–765, 1995.

[10] I. alderón-Ramos, R. D. Morales, R. Servín-Castañeda, A. Pérez-Alvarado, S. García-Hernández, J. De Jesús Barreto, and S. A. Arreola-Villa. Modeling study of turbulent flow in a continuous casting slab mold comparing three ports sen designs. *ISIJ International*, **59**(1):76–85, 2019.

[11] L. Zhang, Y. Wang, and X. Zuo. Flow transport and inclusion motion in steel continuous-casting mold under submerged entry nozzle clogging condition. *Metallurgical and Materials Transactions B: Process Metallurgy and Materials Processing Science*, **39**(4):534–550, 2008.

[12] B. Zhao, B. G. Thomas, S. P. Vanka, and R. J. O'Malley. Transient fluid flow and superheat transport in continuous casting of steel slabs. *Metallurgical and Materials Transactions B: Process Metallurgy and Materials Processing Science*, **36**(6):801–823, 2005.

[13] Huang, X., Thomas, B. G., and F. M. Najjar. Modeling superheat removal during continuous casting of steel slabs. *Metallurgical Transactions B*, **23**(3):339–356, 1992.

[14] J. Sengupta, B. G. Thomas, H. J. Shin, G. G. Lee, and S. H. Kim. Mechanism of hook formation during continuous casting of ultra-low carbon steel slabs. *Metall. Mater. Trans. A.*, **37A**(5):1597, 2006.

[15] S. Wang, X. Zhang, L. Zhang, and Q. Wang. Influence of electromagnetic brake on hook growth and inclusion entrapment beneath the surface of low-carbon continuous casting slabs. *Steel Res. Int.*, **89**: 1–8, 2018.

[16] J. Herbertson, Q.L. He, P.J. Flint, and R.B. Mahapatra. Modelling of metal delivery to continuous casting moulds. *Steelmaking Conf. Proc.*, 1991.

[17] Q. Yuan, B.G. Thomas, and S.P. Vanka. Study of transient flow and particle transport in continuous steel caster molds: Part i. fluid flow. *Metall. Mater. Trans. B*, **35B**:703–714, 2004.

[18] Q. Yuan. *PhD Thesis*. University of Illinois at Urbana-Champaign, 2004.

[19] S. Mahmood. *MS Thesis*. University of Illinois at Urbana-Champaign, 2006.

[20] L. Zhang and Y. Wang. Modeling the entrapment of nonmetallic inclusions in steel continuous-casting billets. *JOM*, **64**:1063–74, 2012.

[21] Z. Liu, B. Li, and M. Jiang. Transient asymmetric flow and bubble transport inside a slab continuous-casting mold. *Metall. Mater. Trans. B*, **45B**:675–97, 2014.

[22] W. Chen, Y. Ren, and L. Zhang. Large eddy simulation on the fluid flow, solidification and entrapment of inclusions in the steel along the full continuous casting slab strand. *JOM*, **70**(12): 2968–2979, 2018.

[23] Q. Yuan and B.G. Thomas. *Third Int Congr. Sci. Technol. Steel- mak.*, pages 745–62, 2005.

[24] K. Jin, B. G. Thomas, and X. Ruan. Modeling and measurements of multiphase flow and bubble entrapment in steel continuous casting. *Metallurgical and Materials Transactions B: Process Metallurgy and Materials Processing Science*, **47**(1), 2016.

[25] S.-M. Cho, B.G. Thomas, J.-Y. Hwang, J.-G. Bang, ans I.-S. Bae. Modeling of inclusion capture in a steel slab caster with vertical section and bending. *Metals*, page 654, 2021.

[26] W. Chen, L. Zhang, Y. Wang, Y. Ren, Q. Ren, and W. Yang. Prediction on the three-dimensional spatial distribution of the number density of inclusions on the entire cross section of a steel continuous casting slab. *International Journal of Heat and Mass Transfer*, **190**, 2022.

[27] K. Jin, S. P. Vanka, and B. G. Thomas. Large eddy simulations of argon bubble transport and cpature in mold region of a continuous steel caster. *TFEC2017*, 2017.

[28] A. Yamauchi, S. Itoyama, Y. Kishimoto, and T. Watanabe. Modeling and measurements of multiphase flow and bubble entrapment in steel continuous casting. *ISIJ international*, **42**:1094–1102, 2002.

[29] H. Yamamura, Y. Mizukami, and K. Misawa. *ISIJ international*, **Supplement**:S223–226, 1996.

[30] H. Nakato, T. Nozaki, Y. Habu, H. Oka, T. Ueda, Y. Kitano, and T. Koshikawa. Improvement of surface quality of continuous cast slabs by high frequency mold oscillation. *Steelmaking Conference Proceedings*, **68**:361–365, 1985.

[31] M. M. Wolf and W. Kurz. Solidification of steel in continuous casting molds. *Solidification and Casting of Metals*, pages 287–294, 1977.

[32] G.-G. Lee, H.-J. Shin, S.-H. Kim, S.-K. Kim, W.-Y. Choi, and B. G. Thomas. Prediction and control of subsurface hooks in continuous cast ultra-low-carbon steel slabs. *Ironmaking and Steelmaking*, **36** (1):39–49, 2009.

[33] K. Jin. *PhD Thesis*. University of Illinois at Urbana-Champaign, 2016.

[34] S.M. Cho and B. G. Thomas. Electromagnetic forces in continuous casting of steel slabs. *Metals*, 2019.

[35] J. Nakashima, J. Fukuda, A. Kiyose, T. Kawase, Y. Ohtani, and M. Doki. Improvement of slab surface quality with in-mold electromagnetic stirring. *Nippon Steel and Sumitomo Metal Corporation*, pages 61–67, 2002.

[36] K. Fujisaki. In-mold electromagnetic stirring in continuous casting. *IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS*, pages 1098–1104, 2001.

[37] C. Pfeiler, M. Wu, and A. Ludwig. Influence of argon gas bubbles and non-metallic inclusions on the flow behavior in steel continuous casting. *Materials Science and Engineering A*, pages 115–120, 2005.

[38] B. G. Thomas and X. Huang. Effect of argon gas on fluid flow in a continuous slab casting mold. *Iron and Steel Society*, pages 273—-289, 1993.

[39] B. G. Thomas, X. Huang, and R. C. Sussman. Simulation of argon gas flow effects in a continuous slab caster. *Metall. Mater. Trans. B*, pages 527—-547, 1994.

[40] B. Li, T. Okane, and T. Umeda. Modeling of molten metal flow in a continuous casting process considering the effects of argon gas injection and static magnetic-field application. *Metall. Mater. Trans. B*, pages 1491–1503, 2000.

[41] Z. Liu, B. Li, M. Jiang, and F. Tsukihashi. Modeling of transient two-phase flow in a continuous casting mold using euler- euler large eddy simulation scheme. *ISIJ Int.*, pages 484–492, 2013.

[42] R. Sanchez-Perez, L. Garcı´a-Demedices, J.P. Ramos, M. Diaz-Cruz, and R.D.Morales. *Metall.Mater. Trans. B*, pages 85–99, 2004.

[43] B.G. Thomas, A. Denissov, and H. Bai. *80th Steelmaking Conference Proceedings, Iron and Steel Society*, pages 375–845, 1997.

[44] H. Bai and B.G. Thomas. *Metall. Mater. Trans. B*, pages 253–67, 2001.

[45] Y. Wang and L. Zhang. *Metall. Mater. Trans. B*, pages 1319–51, 2011.

[46] B.G. Thomas, Q. Yuan, S. Mahmood, R. Liu, and R. Chaudhary. *Metall. Mater. Trans. B*, pages 22–35, 2014.

[47] R.C. Sussman, M.T. Burns, X. Huang, and B.G. Thomas. *Iron and Steel Society*, pages 291–304, 1992.

[48] C. Pfeiler, B.G. Thomas, M. Wu, A. Ludwig, and A. Kharicha. *Steel Res. Int.*, pages 599–607, 2008.

[49] C. Liu, Z. Luo, T. Zhang, D. Shen, W. Nan, and Z. Zou. *J. Iron Steel Res. Int.*, pages 403–07, 2014.

[50] V. Singh, S.K. Ajmani, A.R. Pal, S.K. Singh, and M.B. Denys. Single strand continuous caster tundish furniture comparison for optimal performance. *Ironmak. Steelmak.*, pages 171–179, 2012.

[51] R. Chaudry, C. Ji, B. G. Thomas, and S. P. Vanka. *Metall. Mater. Trans. B*, page 987, 2011.

[52] D.-Y. Sheng and P.G. Jönsson. Effect of thermal buoyancy on fluid flow and residence-time distribution in a single-strand tundish. *Materials (Basel)*, page 1906, 2021.

[53] S. Joo, J.W. Han, and R.I.L. Guthrie. Inclusion behavior and heat-transfer phenomena in steelmaking tundish operations: Part ii. mathematical model for liquid steel in tundishes. *Metall Mater Trans B*, pages 767–777, 1993.

[54] A. Vargas-Zamora, et al. Heat and mass transfer of a convective-stratified flow in a trough type tundish. *International Journal of Heat and Mass Transfer*, pages 3029–3039, 2003.

[55] H. Sun, B. Yan, and J. Zhang. Effect of thermal buoyancy force on the flow, temperature distribution and residence time distribution of molten steel in the slab casting tundish. *The Minerals, Metals, &Materials Society*, 2012.

[56] A. Tripathi, et al. Numerical investigation of fluid flow and heat transfer phenomenon inside a single strand tundish of slab caster. *Steel Res. Int.*, pages 1558–1573, 2015.

[57] O.S. Delgado Ramirez. Thermal and fluid-dynamic optimization of a five strand asymmetric delta shaped billet caster tundish. *Steel Res. Int.*, 2018.

[58] F. Xing, S. G. Zheng, Z.H. Liu, and M.Y. Zhu. Flow field, temperature field, and inclusion removal in a new induction heating tundish with bent channels. *Metals*, page 561, 2019.

[59] H. Tang. Hydrodynamic modeling and mathematical simulation on flow field and inclusion removal in a seven-strand continuous casting tundish with channel type induction heating. *Metals*, page 374, 2018.

[60] S. Chakraborty and Y. Sahai. Stream temperature on the melt flow and continuous casting tundishes. *ISIJ Int.*, pages 960–967, 1991.

[61] Tian-peng Qu, Cheng-jun Liu, and Mao fa Jiang. Numerical simulation for effect of inlet cooling rate on fluid flow and temperature distribution in tundish. *Journal of Iron and Steel Research, International*, pages 12–19, 2012.

[62] S. Chatterjee and K. Chattopadhyay. Transient steel quality under non-isothermal conditions in a multi-strand billet caster tundish: part i. analysis of fluid flow, thermal behaviour and inclusion behaviour. *Ironmaking & Steelmaking Processes, Products and Applications*, pages 403–412, 2017.

[63] K.G. Rackers and B.G. Thomas. Clogging in continuous casting nozzle. *78th Steelmaking Conference Proceeding*, 1995.

[64] H. Olia, H. Yang, S.M. Cho, M. Liang, L. Das, and B. G. Thomas. Pressure distribution and flow rate behavior in continuous-casting slide-gate systems: Pfsg. *Metall. Mater. Trans. B*, pages 1661–1680, 2022.

[65] H. Bai, and B.G. Thomas. Bubble formation during horizontal gas injection into downwards-flowing liquid. *Metall. Mater. Trans. B*, pages 1143–1159, 2001.

[66] Y. Meng and B.G. Thomas. Heat-transfer and solidification model of continuous slab casting: Con1d. *Metallurgical and Materials Transactions B*, pages 685–705, 2003.

[67] Ansys fluent 14.5-theory guide. 2012.

[68] R. Liu. Modeling transient multiphase flow and mold top surface behavior in steel continuous casting. 2014.

[69] A. Tomiyama, I. Kataoka, I. Zun, and T. Sakaguchi. Drag coefficients of single bubbles under normal and micro gravity conditions. *JSME International Journal Series B*, pages 472–479, 1998.

[70] H. Yang. Pressure-energy model of tundish, slide-gate and nozzle flow and application to avoiding aspiration. *CCC Annual Meeting Report*.

[71] H. Bai and B.G. Thomas. Turbulent flow of liquid steel and argon bubbles in slide-gate tundish nozzles: Part ii. effect of operation conditions and nozzle design. *Metall. Mater. Trans. B*, pages 269–284, 2001.

[72] P.H. Dauby. Tcontinuous casting: make better steel and more of it. *Revue de Métallurgie*, 2012.

[73] B. G. Thomas. Chapter 14. in making, shaping and treating of steel. *AISE*, 2003.

[74] S. M. Cho, B. G. Thomas, and S.H. Kim. Effect of nozzle port angle on transient flow and surface slag behavior during continuous steel-slab casting. *Metall. Mater. Trans. B*, pages 52–76, 2018.

[75] S. M. Cho, B. G. Thomas, and S.H. Kim. Transient two-phase flow in slide-gate nozzle and mold of continuous steel slab casting with and without double-ruler electro-magnetic braking. *Metall. Mater. Trans. B*, pages 3080–3098, 2016.

[76] Y. Yin, J. Zhang, S. Lei, and Q. Dong. Numerical study on the capture of large inclusion in slab continuous casting with the effect of in-mold electromagnetic stirring. *ISIJ International*, pages 2165–2174, 2017.

[77] X. Wang, Q. Lai, Z. Luo, Q. Hou, T. Zhang and Z. Zou. Numerical study of inclusion removal in steel continuous casting mold considering interactions between bubbles and inclusions. *ISIJ International*, pages 2062–2070, 2018.

[78] Z. Liu, and B. Li. Transient motion of inclusion cluster in vertical-bending continuous casting caster considering heat transfer and solidification. *Powder Technology*, pages 315–329, 2016.

[79] W. Yang Z. Luo, N. Zhao, and Z. Zou. Numerical analysis of effect of initial bubble size on captured bubble distribution in steel continuous casting using euler-lagrange approach considering bubble coalescence and breakup. *Metals*, page 1160, 2020.

[80] W. Yang Z. Luo, Z. Zou, C. Zhao and Y. You. Modelling and analysis of bubble entrapment by solidification shell in steel continuous casting considering bubble interaction with a coupled cfd-dbm approach. *Powder Technology*, pages 387–400, 2021.

[81] G.G. Lee, B.G. Thomas, S.H. Kim, H.J. Shin, S.K. Baek, C.H. Choi, D.S. Kim, and S.J. Yu. Microstructure near corners of continuous-cast steel slabs showing three-dimensional frozen meniscus and hooks. *Acta Materialia*, pages 6705–6712, 2007.

[82] J. Sengupta, H. J. Shin, B. G. Thomas, and S.H. Kim. Micrograph evidence of meniscus solidification and sub-surface microstructure evolution in continuous-cast ultralow-carbon steels. *Acta Materialia*, pages 1165–1173, 2005.

[83] T. Shi. Effect of argon injection on fluid flow and heat transfer in the continuous slab casting mold.

[84] S. Mishra and R. Datta. Embrittlement of steel. *Encyclopedia of Materials: Science and Technology*, pages 2761–2768, 2001.

[85] T. Emi. 13 - improving steelmaking and steel properties. *Fundamentals of Metallurgy, Woodhead Publishing Series in Metals and Surface Engineering*, pages 503–554, 2005.

[86] Seppo Louhenkilpi. Continuous casting of steel, 2014.

[87] W. Chen, Y. Ren, L. Zhang, and P.P. Scheller. Numerical simulation of steel and argon gas two-phase flow in continuous casting using les + vof + dpm model. *JOM*, pages 1158–68, 2019.

[88] Z. Liu and B. Li. Large-eddy simulation of transient horizontal gas–liquid flow in continuous casting using dynamic subgrid-scale model. *Metallurgical and Materials Transactions B*, pages 1833–1849, 2017.

[89] X. Li, B. Li, Z. Liu, R. Niu, Y. Liu, C. Zhao, C. Huang, H. Qiao, and T. Yuan. Large eddy simulation of multi-phase flow and slag entrapment in a continuous casting mold. *Metals*, pages 273–289, 2019.

[90] B.G. Thomas, and X. Huang. Effect of argon gas on fluid flow in a continuous slab casting mold. *76 th Steelmaking Conference*, pages 273–289, 2014.

[91] R. Sánchez-Pérez, L. García-Demedices, J. Palafox. Ramos, M. Díaz-Cruz, and R.D. Morales. Dynamics of coupled and uncoupled two-phase flows in a slab mold. *Metallurgical and Materials Transactions B*, pages 85–99, 2004.

[92] R.C. Sussman, M.T. Burns, X. Huang, and B.G. Thomas. *Iron Steelmak.*, pages 14–16, 1993.

[93] P. Rosin. Laws governing the fineness of powdered coal. *Journal of Institute of Fuel*, 1993.

[94] G.G. Lee, B.G. Thomas, and S.H. Kim. Effect of refractory properties on initial bubble formation in continuous-casting nozzles. *Metals and Materials International*, pages 501–506, 2010.

[95] B.E. Launder and D.B. Spalding. The numerical computation of turbulent flows. *Comput. Methods Appl. Mech. Eng.*, pages 269–289, 1974.

[96] A. Akhtar. Analysis of the nail board experiment to measure the liquid slag depth in steel continuous casting. 2016.

[97] H.-J. Shi, S.-H. Kim, B. G. Thomas, G.-G. Lee, J.-M. Park, and J. Sengupta. Measurement and prediction of lubrication, powder consumption, and oscillation mark profiles in ultra-low carbon steel slabs. *ISIJ international*, pages 1635–1644, 2006.

[98] S. A. Moris. An investigation of particle trajectories in two-phase flow systems. *Journal of Fluid Mechanics*, pages 193–208, 1972.

[99] C. Crowe, M. Sommerfield, and Y. Tsuji. Multiphase flows with droplets and particles. *CRC Press, Boca Raton*, 1998.

[100] Q. Wang, K.D. Squires, M. Chen, and J.B. McLaughlin. On the role of the lift force in turbulence simulations of particle deposition. *Int. J. Multiphase Flow*, pages 749–763, 1997.

[101] R. Mei. Brief communication: An approximate expression for the shear lift force on a spherical particle at finite reynolds number. *International Journal of Multiphase Flow*, pages 145–147, 1992.

[102] D.R. Uhlmann and B. Chalmers. Interaction between particles and a solid-liquid interface. *Journal of Applied Physics*, pages 2986–2993, 1964.

[103] D. Shangguan, S. Ahuja, and D.M. Stefanescu. An analytical model for the interaction between an insoluble particle and an advancing solid/liquid interface. *Metallurgical Transactions A*, pages 669–680, 1992.

[104] J. Potschke and V. Rogge. On the behavior of foreign particle at an advancing solid-liquid interface. *Journal of Crystal Growth*, pages 726–738, 1989.

[105] G. Kaptay and K.K. Kelemen. The force acting on a sphere moving towards a solidification front due to an interfacial energy gradient at the sphere/liquid interface. *ISIJ International*, pages 305–307, 2001.

[106] B.G. Thomas and R.J. O'Malley. Measurement of temperature, solidification, and microstructure in a continuous cast thin slab. *MCWASP VIII, San Diego, CA, June 7-12*, 1998.

[107] M. El-Bealy and B.G. Thomas. Prediction of dendrite arm spacing for low alloy steel casting processes. *Metallurgical and Materials Transactions B*, pages 689–693, 1996.

[108] J. Miettinen, V.K.d. Barcellos, V.L.d.S. Gschwenter, C.A.d. Santos, S. Louhenkilpi, H. Kytönen, and J.A. Spim. Modelling of heat transfer, dendrite microstructure and grain size in continuous casting of steels. *Steel Research International*, pages 461–471, 2010.

[109] H. Mehrara, B. Santillana, D.G. Eskin, R. Boom, L. Katgerman, and G. Abbel. Modeling of primary dendrite arm spacing variations in thin-slab casting of low carbon and low alloy steels. *IOP Conference Series: Materials Science and Engineering*, pages 1–7, 2011.

[110] Y. Ueshima, S. Mizoguchi, T. Matsumiya, and H. Kajioka. Analysis of solute distribution in dendrites of carbon steel with δ/γ transformation during solidification. *Metallurgical Transactions B*, pages 845–859, 1986.

[111] J.A. Dantzig and C.L. Tucker. Heat conduction and materials processing. *Modeling in Materials Processing, Cambridge, United Kingdom, U.K., Cambridge university press*, pages 113–120, 2001.

[112] W. Kurz and D.J. Fisher. Dendrite growth at the limit of stability: tip radius and spacing. *Acta Metallurgica*, pages 11–20, 1981.

[113] S.V. Patankar. Numerical heat transfer and fluid flow. *CRC Press*, 1980.

[114] A. Ghobadian and S. A. Vasquez. A general purpose implicit coupled algorithm for the solution of eulerian multiphase transport equation". international conference on multiphase flow. 2007.

[115] Chaudhary, R., Lee, G., Thomas, B. G., and S. Kim. Transient mold fluid flow with well- and mountain-bottom nozzles in continuous casting of steel. *Metallurgical and Materials Transactions B*, pages 870–884, 2008.

[116] R. Chaudhary. *PhD Thesis*. University of Illinois at Urbana-Champaign, 2001.

[117] H. Yang, S. P. Vanka, and B. G. Thomas. A hybrid eulerian-eulerian discrete-phase model of turbulent bubbly flow. *Journal of Fluids Engineering, Transactions of the ASME*, pages 1–12, 2018.

[118] J. Smagorinsky. General circulation experiments with the primitive equations. i. the basic experiment. *Month. Wea. Rev.*, pages 99–164, 1963.

[119] X. Sun, B. Li, H. Lu, Y. Zhong, Z. Ren, and Z. Lei. Steel / slag interface behavior under multifunction electromagnetic driving in a continuous casting slab mold. *Metals*, 2019.

[120] Y. Miki and B. G. Thomas. Modeling of inclusion removal in a tundish. *Metallurgical and Materials Transactions B*, pages 639–654, 1999.

[121] F. R. Menter. Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA Journal*, pages 1598–1605, 1994.

[122] F.R. Menter, M. Kuntz, and R. Langtry. Ten years of industrial experience with the sst turbulence model. *Turbulence, heat and mass transfer*, pages 625–632, 2003.

[123] H. Versteeg and W. Malalasekra. *An Introduction to Computational Fluid Dynamics - the Finite Volume Method, 2nd edition*. Pearson Eduction, Ltd., 2007.

APPENDIX A

USER DEFINED FUNCTIONS

The followings are the UDFs used for the pressure outlet, the mass / momentum / energy sink terms, and the particle hook / entrapment / engulfment capture mechanisms in the simulations of this thesis:

Listing A.1: Ferrostatic Pressure on the Outlet of Nozzle and Strand Domain

```
#include "udf.h"
#define dens  7000 /* steel density in kg/m^3*/
#define grav  9.81 /* gravitational acceleration in m/s^3 */


DEFINE_PROFILE(pressure_profile,t,i)
{
    real x[ND_ND];     /* this will hold the position vector */
    real y;            /* y-coordinate. Its absolute value is the vertical distance
        below the meniscus */
    face_t f;
    begin_f_loop(f,t)  /* loop through all outlet faces */
      {
       F_CENTROID(x,f,t);
       y = x[1];
       F_PROFILE(f,t,i) = - dens * y * grav;  /* Ferrostatic pressure */
      }
    end_f_loop(f,t)
}
```

Listing A.2: Argon Gas Mass / Momentum / Enegy Sinks into the Top Surface

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

#define top_surface   19      /* Top surface ID number */

double rho_argon = 0.416;    /* argon density in kg/m^3 */
double cp_ar = 520;             /* argon heat capacity in J/(kgK)*/
double T_ref = 298.15;          /* Ambient reference temperature for Argon in K */



/* ////////////////////////////////////////////////////////////// Mass sink
   ////////////////////////////////////////////////////////////// */

DEFINE_SOURCE(argon_mass_sink, c, t, dS, eqn)     /* name, cell, time, derivative of
    the source, eqn number */

{
        double source, vy, area_facey;
```

```
        double A[ND_ND];                /* area vector */
        int i;

        face_t f;                       /* An integer data type that identifies a
            particular face within a face thread : tiny cell face */

        Thread *tf;                     /* A structure data type that stores data that
            is common to the group of cells or faces that it represents : boundary we
             define in FLUENT (wall, inlet, outlet...) */

        source = 0.;
        dS[eqn] = 0.;                   /* Array that contains the derivative of the
            source term with respect to the dependent variable of the transport
            equation. */


        c_face_loop(c,t,i)
        {
                f = C_FACE(c, t, i);            /* return the global face index face_t
                     f for the given cell_t c, Thread *t, and local face index number
                     i : get global face index face_t f */
                tf = C_FACE_THREAD(c, t, i);  /* return the Thread *t of the face_t
                    f that is returned by C_FACE  : get Thread *t of the face_t f */
                 if (THREAD_ID(C_FACE_THREAD(c, t, i)) == top_surface)
                 {
                        vy = C_V(c,t);     /* vertical velocity of argon gas */
                        if(vy > 0)          /* gas mass sink only happens when the
                            gas vertical velocity is towards the top surface*/
                        {

                        F_AREA(A, f, tf);
                                                                              /*
                            F_AREA: generates area vector A[ND_ND] as an output */
                        area_facey = fabs(A[1]);

                        source = - rho_argon * C_V(c,t) * area_facey * C_VOF(c,t) /
                            C_VOLUME(c, t);              /* mass sink = rho *
                             Velocity * Area * fraction / Volume, in the unit of kg/(m
                             ^3s)  */
                    dS[eqn] = 0;
                        }
        }
        }
        return source;
        return dS[eqn];
}


/************************************************************ Momentum sink
    ***********************************************/

DEFINE_SOURCE(argon_x_mom_sink, c, t, dS, eqn)
{
        double source, vy, area_facey;
        double A[ND_ND];                                        /* area vector */
        int i;

        face_t f;                       /* An integer data type that identifies a
            particular face within a face thread : tiny cell face */
```

```
        Thread *tf;                     /* A structure data type that stores data that
            is common to the group of cells or faces that it represents : boundary we
             define in FLUENT (wall, inlet, outlet...) */

        source = 0.;
        dS[eqn] = 0.;                   /* Array that contains the derivative of the
            source term with respect to the dependent variable of the transport
            equation. */


        c_face_loop(c,t,i)
        {
                f = C_FACE(c, t, i);            /* return the global face index face_t
                     f for the given cell_t c, Thread *t, and local face index number
                     i : get global face index face_t f */
                tf = C_FACE_THREAD(c, t, i);  /* return the Thread *t of the face_t
                    f that is returned by C_FACE  : get Thread *t of the face_t f */
                 if (THREAD_ID(C_FACE_THREAD(c, t, i)) == top_surface)
                 {
                        vy = C_V(c,t);  /* vertical velocity of argon gas */
                        if(vy > 0)        /* gas mass sink only happens when the gas
                            vertical velocity is towards the top surface*/
                        {

                        F_AREA(A, f, tf);
                                                                              /*
                            F_AREA: generates area vector A[ND_ND] as an output */
                        area_facey = fabs(A[1]);

                        source = - rho_argon * C_V(c,t) * area_facey * C_VOF(c,t) /
                            C_VOLUME(c, t) * C_U(c,t);              /* x-mom sink =
                            mass_sink * x-velocity, in the unit of kg/(m^3s)  */
                    dS[eqn] = 0;
                        }
        }
        }
        return source;
        return dS[eqn];
}


DEFINE_SOURCE(argon_y_mom_sink, c, t, dS, eqn)
{
        double source, vy, area_facey;
        double A[ND_ND];                                        /* area vector */
        int i;

        face_t f;       /* An integer data type that identifies a particular face
            within a face thread : tiny cell face*/

        Thread *tf;     /*A structure data type that stores data that is common to
            the group of cells or faces that it represents : boundary we define in
            FLUENT (wall, inlet, outlet...)*/

        source  = 0.;
        dS[eqn] = 0.;    /* Array that contains the derivative of the source term
            with respect to the dependent variable of the transport equation.*/
```

```
        c_face_loop(c,t,i)
        {
                f = C_FACE(c, t, i);          /* return the global face index face_t
                     f for the given cell_t c, Thread *t, and local face index number
                     i : get global face index face_t f */
                tf = C_FACE_THREAD(c, t, i); /* return the Thread *t of the face_t
                    f that is returned by C_FACE  : get Thread *t of the face_t f */
                 if (THREAD_ID(C_FACE_THREAD(c, t, i)) == top_surface)
                 {
                        vy = C_V(c,t);    /* vertical velocity of argon gas */
                        if(vy > 0)         /* gas mass sink only happens when the gas
                             vertical velocity is towards the top surface*/
                        {

                        F_AREA(A, f, tf);
                                                                              /*
                            F_AREA: generates area vector A[ND_ND] as an output */
                        area_facey = fabs(A[1]);

                        source = - rho_argon * C_V(c,t) * area_facey * C_VOF(c,t) /
                            C_VOLUME(c, t) * C_V(c,t);              /* y-mom sink =
                             mass_sink * y-velocity, in the unit of kg/(m^3s)  */
                    dS[eqn] = 0;
                        }
        }
        }
           return source;
           return dS[eqn];
}




DEFINE_SOURCE(argon_z_mom_sink, c, t, dS, eqn)
{
        double source, vy, area_facey;
        double A[ND_ND];                                        /* area vector */

        int i;

        face_t f;                       /* An integer data type that identifies a
            particular face within a face thread : tiny cell face */

        Thread *tf;                     /* A structure data type that stores data that
            is common to the group of cells or faces that it represents : boundary we
             define in FLUENT (wall, inlet, outlet...) */

        source = 0.;
        dS[eqn] = 0.;                   /* Array that contains the derivative of the
            source term with respect to the dependent variable of the transport
            equation. */


        c_face_loop(c,t,i)
        {
                f = C_FACE(c, t, i);          /* return the global face index face_t
                     f for the given cell_t c, Thread *t, and local face index number
                     i : get global face index face_t f */
```

```
                tf = C_FACE_THREAD(c, t, i);  /* return the Thread *t of the face_t
                    f that is returned by C_FACE  : get Thread *t of the face_t f */
                 if (THREAD_ID(C_FACE_THREAD(c, t, i)) == top_surface)
                 {
                        vy = C_V(c,t);    /* vertical velocity of argon gas */
                        if(vy > 0)         /* gas mass sink only happens when the gas
                             vertical velocity is towards the top surface*/
                        {

                        F_AREA(A, f, tf);
                                                                              /*
                            F_AREA: generates area vector A[ND_ND] as an output */
                        area_facey = fabs(A[1]);

                        source = - rho_argon * C_V(c,t) * area_facey * C_VOF(c,t) /
                            C_VOLUME(c, t) * C_W(c,t);              /* z-mom sink =
                             mass_sink * z-velocity, in the unit of kg/(m^3s)  */
                    dS[eqn] = 0;
                        }
        }
        }
        return source;
        return dS[eqn];
}

DEFINE_SOURCE(argon_energy_sink, c, t, dS, eqn)
{
        double source, vy, area_facey;
        double A[ND_ND];                                        /* area vector */

        int i;

        face_t f;                       /* An integer data type that identifies a
            particular face within a face thread : tiny cell face */

        Thread *tf;                     /* A structure data type that stores data that
            is common to the group of cells or faces that it represents : boundary we
             define in FLUENT (wall, inlet, outlet...) */

        source = 0.;
        dS[eqn] = 0.;                   /* Array that contains the derivative of the
            source term with respect to the dependent variable of the transport
            equation. */


        c_face_loop(c,t,i)
        {
                f = C_FACE(c, t, i);          /* return the global face index face_t
                     f for the given cell_t c, Thread *t, and local face index number
                     i : get global face index face_t f */
                tf = C_FACE_THREAD(c, t, i);  /* return the Thread *t of the face_t
                    f that is returned by C_FACE  : get Thread *t of the face_t f */
                 if (THREAD_ID(C_FACE_THREAD(c, t, i)) == top_surface)
                 {
                        vy = C_V(c,t);    /* vertical velocity of argon gas */
                        if(vy > 0)         /* gas mass sink only happens when the gas
                             vertical velocity is towards the top surface*/
                        {
```

```
                        F_AREA(A, f, tf);
                                                                        /*
                            F_AREA: generates area vector A[ND_ND] as an output */
                        area_facey = fabs(A[1]);

                        source = - rho_argon * C_V(c,t) * area_facey * C_VOF(c,t) /
                            C_VOLUME(c, t) * cp_ar * (C_T(c,t) - T_ref);
                            /* energy sink = mass_sink * (T-T_ref) * cp_ar, in the
                            unit of kg/(m^3s)  */
                    dS[eqn] = 0;
                        }
        }
        }
        return source;
        return dS[eqn];
}
```

Listing A.3: Molten Steel Mass / Momentum / Enegy Sinks on Boundary Cells

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

#define dens        7000      /* Steel density; if considering temperature, then C_R(
    c,t) is used. kg/m^3 */
#define vcast       0.01      /* Casting speed 0.6 (m/min) */
#define SL          2.665     /* The straight strand height in m */
#define CurveNumber 8         /* Number of different curvatures in the curved strand
    */
#define SC          0.0032    /* Solidification Constant in m/s^0.5 */

 double cp = 680; /* steel heat capacity in J /(kg * K) */
 double Tref = 298.15; /* reference temperature for energy sink */

 double wide_shell_OR[CurveNumber+1] = {7, 22, 23, 24, 25, 26, 27, 28, 29};      /*
     Thread_ID for shell surface on WF_OR side, the elements are listed from menisucs
      to outlet */
 double wide_shell_IR[CurveNumber+1] = {6, 14, 15, 16, 17, 18, 19, 20, 21};      /*
     Thread ID for shell surface on WF_IR side, the elements are listed from menisucs
      to outlet */
 double narrow_shell[CurveNumber+1] = {8, 30, 31, 32, 33, 34, 35, 36, 37};       /*
     Thread ID for shell surface on NF side, the elements are listed from menisucs to
      outlet */

 double R[CurveNumber] = { 84.85, 41.85, 27.55, 20.35, 16.15, 13.35, 11.35, 9.85 };
                    /* Radius for different curvatures in m */
 double x0[CurveNumber] = {84.85, 41.85, 27.55, 20.35, 16.15, 13.36, 11.36, 9.87};
             /* x xoordinate of the center of each curvature */
 double y_0[CurveNumber] = {-2.665, -2.799, -2.934, -3.071, -3.206, -3.342, -3.479,
    -3.619};          /* y xoordinate of the center of each curvature */
 double CA[CurveNumber]= {0.003118, 0.00631, 0.009567, 0.013171, 0.016564, 0.02,
    0.024348, 0.235}; /* The Central Angle (CA) of each curvature */
```

```
/* ---------------------------------------------------------------------- Mass sink
   ----------------------------------------------------------------------*/

DEFINE_SOURCE(steel_mass_sink_wide_shell_OR, c, t, dS, eqn)     /* name, cell, time,
    derivative of the source, eqn number */

{
        real source, x1, y1, delta_x, delta_y, A_y, A_N, A_pro, dis, theta, CA_0,
            theta_pro;
        int i;
        int a;
        int b;
        int b1;
        real x[ND_ND];      /* coordinate position vector*/
        real A[ND_ND];      /* area vector */
        face_t f;   /* An integer data type that identifies a particular face within
            a face thread : tiny cell face */

        Thread *tf;   /* A structure data type that stores data that is common to
            the group of cells or faces that it represents : boundary we define in
            FLUENT (wall, inlet, outlet...) */

         source  = 0.;
         dS[eqn] = 0.;  /* Array that contains the derivative of the source term
             with respect to the dependent variable of the transport equation. */

         c_face_loop(c,t,i)   /* loops over all faces of a given cell (c: cell) */
         {
                f = C_FACE(c,t,i);   /* return the global face index face_t f for
                    the given cell_t c, Thread *t, and local face index number i :
                    get global face index face_t f */
                tf = C_FACE_THREAD(c,t,i); /* return the Thread *t of the face_t f
                     that is returned by C_FACE  : get Thread *t of the face_t f */

          if(THREAD_ID(C_FACE_THREAD(c,t,i)) == wide_shell_OR[0]) /* if the face is
              on the straight part of the strand */
           {
                  F_AREA(A,f,tf);                                          /* F_AREA:
                      generates area vector A[ND_ND] as an output */
                  A_y = A[1];                                              /*  y
                      component of area vector A[2]  */
                  source = -C_R(c,t)*vcast* fabs(A_y)/C_VOLUME(c,t);       /* C_R(
                      c,t)*vcast* fabs(area_face)/C_VOLUME(c,t) : kg/s/m-3  ->
                      correct unit for continuity */
                  dS[eqn] = -vcast* fabs(A_y)/C_VOLUME(c,t);
           }

          for (a = 1; a <= CurveNumber; a++) /* for-loop through all faces on curved
              WF_OR */
           {
                 if (THREAD_ID(C_FACE_THREAD(c, t, i)) == wide_shell_OR[a])
                 {
                        F_AREA(A, f, tf);                 /* F_AREA: generates area
                            vector A[ND_ND] as an output */
                        F_CENTROID(x, f, tf);                      /* F_CENTROID:
                            generates coordinate position vector x[ND_ND] as an
                            output */
                        x1 = x[0];                        /* x coordinate position of
                            face centroid */
```

```
                y1 = x[1];                          /* y coordinate position of
                    face centroid */

                A_N = NV_MAG(A);                                    /* face
                    area magnitude */

            delta_x = fabs(x1-x0[a-1]);
                delta_y = fabs(y1-y_0[a-1]);
                theta = atan((delta_y)/(delta_x));              /* the
                    angle between the horizontal line (x direction) and the
                    line connecting the face-center point and the acr-center
                    point in radian */

                CA_0 = theta;      /* initialization of the central angle of
                    the arc section the face is in */
                dis = 0;     /* initialization of the the distance below
                    meniscus along the strand in m */

                if (a == 1)
                {
                        dis = SL + CA_0 * R[0];    /* the distance below
                            meniscus along the strand in m */
                }
                else
                {
                        for (b = 0; b <= (a-2); b++)
                        {
                                CA_0 -= CA[b];  /* the central angle of the
                                    arc */
                        }

                        dis = SL + CA_0 * R[a-1];

                        for (b1 = 0; b1 <= (a-2); b1++)
                        {
                                dis += CA[b1] * R[b1];     /* the distance
                                    below meniscus along the strand in m */
                        }

                }

                theta_pro = atan(SC/(2 * sqrt(vcast * dis)));     /* the
                    angle between the face and the casting direction in
                    radian*/
                A_pro = A_N * sin(theta_pro);                   /* the
                    projected face area in the casting direction*/

                source = -C_R(c,t)*vcast*A_pro / C_VOLUME(c, t);      /* C_R(
                    c,t)*vcast* fabs(area_face_cast)/C_VOLUME(c,t) : kg/s/m-3
                      -> correct unit for continuity */
            dS[eqn] = -vcast*A_pro / C_VOLUME(c, t);
        }
            }
        }
         return source;
         return dS[eqn];
}
```

```
DEFINE_SOURCE(steel_mass_sink_wide_shell_IR, c, t, dS, eqn)    /* name, cell, time,
    derivative of the source, eqn number */

{
        real source, x1, y1, delta_x, delta_y, A_y, A_N, A_pro, dis, theta, CA_0,
            theta_pro;
        int i;
        int a;
        int b;
        int b1;
        real x[ND_ND];      /* coordinate position vector*/
        real A[ND_ND];      /* area vector */
        face_t f;   /* An integer data type that identifies a particular face within
            a face thread : tiny cell face */

        Thread *tf;   /* A structure data type that stores data that is common to
            the group of cells or faces that it represents : boundary we define in
            FLUENT (wall, inlet, outlet...) */

         source = 0.;
         dS[eqn] = 0.;  /* Array that contains the derivative of the source term
             with respect to the dependent variable of the transport equation. */

         c_face_loop(c,t,i)   /* loops over all faces of a given cell (c: cell) */
         {
                 f = C_FACE(c,t,i);   /* return the global face index face_t f for
                     the given cell_t c, Thread *t, and local face index number i :
                     get global face index face_t f */
                 tf = C_FACE_THREAD(c,t,i);  /* return the Thread *t of the face_t f
                      that is returned by C_FACE  : get Thread *t of the face_t f */

          if(THREAD_ID(C_FACE_THREAD(c,t,i)) == wide_shell_IR[0]) /* if the face is
              on the straight part of the strand */
           {
                   F_AREA(A,f,tf);                                          /* F_AREA:
                       generates area vector A[ND_ND] as an output */
                   A_y = A[1];                                              /*  y
                       component of area vector A[2]  */
                   source = -C_R(c,t)*vcast* fabs(A_y)/C_VOLUME(c,t);       /* C_R(
                       c,t)*vcast* fabs(area_face)/C_VOLUME(c,t) : kg/s/m-3  ->
                       correct unit for continuity */
                   dS[eqn] = -vcast* fabs(A_y)/C_VOLUME(c,t);
           }

          for (a = 1; a <= CurveNumber; a++) /* for-loop through all faces on curved
              WF_IR */
           {
                 if (THREAD_ID(C_FACE_THREAD(c, t, i)) == wide_shell_IR[a])
                 {
                        F_AREA(A, f, tf);                 /* F_AREA: generates area
                            vector A[ND_ND] as an output */
                        F_CENTROID(x, f, tf);                      /* F_CENTROID:
                            generates coordinate position vector x[ND_ND] as an
                            output */
                        x1 = x[0];                        /* x coordinate position of
                            face centroid */
                        y1 = x[1];                        /* y coordinate position of
                            face centroid */
```

```
                A_N = NV_MAG(A);                                    /* face
                    area magnitude */

            delta_x = fabs(x1-x0[a-1]);
                delta_y = fabs(y1-y_0[a-1]);
                theta = atan((delta_y)/(delta_x));                  /* the
                    angle between the horizontal line (x direction) and the
                    line connecting the face-center point and the acr-center
                    point in radian */

                CA_0 = theta;      /* initialization of the central angle of
                    the arc section the face is in */
                dis = 0;     /* initialization of the distance below the
                    meniscus along the strand in m */

                if (a == 1)
                {
                        dis = SL + CA_0 * R[0];    /* the distance below
                            meniscus along the strand in m */
                }
                else
                {
                        for (b = 0; b <= (a-2); b++)
                        {
                                CA_0 -= CA[b];  /* the central angle of the
                                    arc */
                        }

                        dis = SL + CA_0 * R[a-1];

                        for (b1 = 0; b1 <= (a-2); b1++)
                        {
                                dis += CA[b1] * R[b1];     /* the distance
                                    below meniscus along the strand in m */
                        }

                }

                theta_pro = atan(SC/(2 * sqrt(vcast * dis)));     /* the
                    angle between the face and the casting speed in radian */
                A_pro = A_N * sin(theta_pro);    /* the projected face area
                    in casting direction */

                source = -C_R(c,t)*vcast*A_pro / C_VOLUME(c, t);         /*
                    C_R(c,t)*vcast* fabs(area_face_cast)/C_VOLUME(c,t) : kg/s
                    /m^-3  -> correct unit for continuity */
            dS[eqn] = -vcast*A_pro / C_VOLUME(c, t);
        }
            }
        }
         return source;
         return dS[eqn];
}

DEFINE_SOURCE(steel_mass_sink_narrow_shell, c, t, dS, eqn)     /* name, cell, time,
    derivative of the source, eqn number */

{
```

```
real source, x1, y1, delta_x, delta_y, A_y, A_N, A_pro, dis, theta, CA_0,
    theta_pro;
int i;
int a;
int b;
int b1;
real x[ND_ND];      /* coordinate position vector*/
real A[ND_ND];      /* area vector */
face_t f;   /* An integer data type that identifies a particular face within
     a face thread : tiny cell face */

Thread *tf;   /* A structure data type that stores data that is common to
    the group of cells or faces that it represents : boundary we define in
    FLUENT (wall, inlet, outlet...) */

 source  = 0.;
 dS[eqn] = 0.;  /* Array that contains the derivative of the source term
     with respect to the dependent variable of the transport equation. */

 c_face_loop(c,t,i)   /* loops over all faces of a given cell (c: cell) */
 {
         f = C_FACE(c,t,i);   /* return the global face index face_t f for
             the given cell_t c, Thread *t, and local face index number i :
             get global face index face_t f */
         tf = C_FACE_THREAD(c,t,i);  /* return the Thread *t of the face_t f
              that is returned by C_FACE  : get Thread *t of the face_t f */

  if(THREAD_ID(C_FACE_THREAD(c,t,i)) == narrow_shell[0]) /* if the face is
     on the straight part of the strand */
  {
           F_AREA(A,f,tf);                                          /* F_AREA:
               generates area vector A[ND_ND] as an output */
           A_y = A[1];                                              /*  y
               component of area vector A[2]  */
           source = -C_R(c,t)*vcast* fabs(A_y)/C_VOLUME(c,t);       /* C_R(
               c,t)*vcast* fabs(area_face)/C_VOLUME(c,t) : kg/s/m^-3  ->
               correct unit for continuity */
           dS[eqn] = -vcast* fabs(A_y)/C_VOLUME(c,t);
  }

  for (a = 1; a <= CurveNumber; a++) /* for-loop through all faces on curved
      NF */
  {
        if (THREAD_ID(C_FACE_THREAD(c, t, i)) == narrow_shell[a])
        {
              F_AREA(A, f, tf);                 /* F_AREA: generates area
                  vector A[ND_ND] as an output */
              F_CENTROID(x, f, tf);                    /* F_CENTROID:
                  generates coordinate position vector x[ND_ND] as an
                  output */
              x1 = x[0];                        /* x coordinate position of
                  face centroid */
              y1 = x[1];                        /* y coordinate position of
                  face centroid */

              A_N = NV_MAG(A);                                  /* face
                  area magnitude */

       delta_x = fabs(x1-x0[a-1]);
```

```
                        delta_y = fabs(y1-y_0[a-1]);
                        theta = atan((delta_y)/(delta_x));                  /* the
                            angle between the horizontal line (x direction) and the
                            line connecting the face-center point and the acr-center
                            point in radian */

                        CA_0 = theta;      /* initialization of the central angle of
                            the arc section the face is in */
                        dis = 0;     /* initialization of the distance below the
                            menidcus along the strand in m */

                        if (a == 1)
                        {
                                dis = SL + CA_0 * R[0];    /* the distance below
                                    meniscus along the strand in m */
                        }
                        else
                        {
                                for (b = 0; b <= (a-2); b++)
                                {
                                        CA_0 -= CA[b];  /* the central angle of the
                                            arc */
                                }

                                dis = SL + CA_0 * R[a-1];

                                for (b1 = 0; b1 <= (a-2); b1++)
                                {
                                        dis += CA[b1] * R[b1];     /* the distance
                                            below meniscus along the strand in m */
                                }

                        }

                        theta_pro = atan(SC/(2 * sqrt(vcast * dis)));     /* the
                            angle between the face and the casting direction in
                            radian */
                        A_pro = A_N * sin(theta_pro);    /* the projected face area
                            in casting direction */

                        source = -C_R(c,t)*vcast*A_pro / C_VOLUME(c, t);         /*
                            C_R(c,t)*vcast* fabs(area_face_cast)/C_VOLUME(c,t) : kg/s
                            /m^-3  -> correct unit for continuity */
                    dS[eqn] = -vcast*A_pro / C_VOLUME(c, t);
                }
                }
                }
                 return source;
                 return dS[eqn];
}

/* ---------------------------------------------------------------- Momentum sink
    ----------------------------------------------------------------*/
DEFINE_SOURCE(steel_xmom_sink_wide_shell_OR, c, t, dS, eqn)     /* name, cell, time,
    derivative of the source, eqn number */

{
        real source, x1, y1, delta_x, delta_y, A_y, A_N, A_pro, dis, theta, CA_0,
            theta_pro;
```

```
int i;
int a;
int b;
int b1;
real x[ND_ND];      /* coordinate position vector*/
real A[ND_ND];      /* area vector */
face_t f;   /* An integer data type that identifies a particular face within
    a face thread : tiny cell face */

Thread *tf;   /* A structure data type that stores data that is common to
   the group of cells or faces that it represents : boundary we define in
   FLUENT (wall, inlet, outlet...) */

 source  = 0.;
 dS[eqn] = 0.;  /* Array that contains the derivative of the source term
     with respect to the dependent variable of the transport equation. */

 c_face_loop(c,t,i)   /* loops over all faces of a given cell (c: cell) */
 {
         f = C_FACE(c,t,i);   /* return the global face index face_t f for
             the given cell_t c, Thread *t, and local face index number i :
             get global face index face_t f */
         tf = C_FACE_THREAD(c,t,i);  /* return the Thread *t of the face_t f
              that is returned by C_FACE  : get Thread *t of the face_t f */

  if(THREAD_ID(C_FACE_THREAD(c,t,i)) == wide_shell_OR[0]) /* if the face is
     on the straight part of the strand */
   {
           F_AREA(A,f,tf);                                          /* F_AREA:
               generates area vector A[ND_ND] as an output */
           A_y = A[1];                                              /*  y
               component of area vector A[2]  */
           source = -C_R(c,t)*vcast* fabs(A_y) * C_U(c,t) /C_VOLUME(c,t);
                    /* C_R(c,t)*vcast* fabs(area_face_cast)*vx/C_VOLUME(c,t)
                : kg/s^-2/m-2  -> correct unit for momentum */
           dS[eqn] = -C_R(c,t)*vcast* fabs(A_y) /C_VOLUME(c,t);
   }

  for (a = 1; a <= CurveNumber; a++) /* for-loop through all faces on curved
      WF_OR */
   {
          if (THREAD_ID(C_FACE_THREAD(c, t, i)) == wide_shell_OR[a])
          {
                 F_AREA(A, f, tf);                  /* F_AREA: generates area
                     vector A[ND_ND] as an output */
                 F_CENTROID(x, f, tf);                      /* F_CENTROID:
                     generates coordinate position vector x[ND_ND] as an
                     output */
                 x1 = x[0];                         /* x coordinate position of
                     face centroid */
                 y1 = x[1];                         /* y coordinate position of
                     face centroid */

                 A_N = NV_MAG(A);                                       /* face
                     area magnitude */

        delta_x = fabs(x1-x0[a-1]);
                 delta_y = fabs(y1-y_0[a-1]);
```

```
                        theta = atan((delta_y)/(delta_x));                /* the
                            angle between the horizontal line (x direction) and the
                            line connecting the face-center point and the acr-center
                            point in radian */

                        CA_0 = theta;      /* initialization of the central angle of
                            the arc section the face is in */
                        dis = 0;     /* initialization of the distance below the
                            meniscus along the strand */

                        if (a == 1)
                        {
                                dis = SL + CA_0 * R[0];    /* the distance below the
                                    meniscus along the strand in m */
                        }
                        else
                        {
                                for (b = 0; b <= (a-2); b++)
                                {
                                        CA_0 -= CA[b];  /* the central angle of the
                                            arc section the face is in */
                                }

                                dis = SL + CA_0 * R[a-1];

                                for (b1 = 0; b1 <= (a-2); b1++)
                                {
                                        dis += CA[b1] * R[b1];     /* the distance
                                            below the meniscus along the strand in m
                                            */
                                }

                        }

                        theta_pro = atan(SC/(2 * sqrt(vcast * dis)));     /* the
                            angle between the face and the casting direction in
                            radian */
                        A_pro = A_N * sin(theta_pro);    /* the projected face area
                            in the casting direction in m^2 */

                        source = -C_R(c,t)*vcast*A_pro * C_U(c,t) / C_VOLUME(c, t);
                                    /* C_R(c,t)*vcast* fabs(area_face_cast)*v_x/
                                C_VOLUME(c,t) : kg/s^-2/m^-2  -> correct unit for
                                continuity */
                    dS[eqn] = -C_R(c,t)*vcast*A_pro / C_VOLUME(c, t);
            }
                }
            }
             return source;
             return dS[eqn];
}

DEFINE_SOURCE(steel_xmom_sink_wide_shell_IR, c, t, dS, eqn)     /* name, cell, time,
    derivative of the source, eqn number */

{
        real source, x1, y1, delta_x, delta_y, A_y, A_N, A_pro, dis, theta, CA_0,
            theta_pro;
        int i;
```

```
int a;
int b;
int b1;
real x[ND_ND];      /* coordinate position vector*/
real A[ND_ND];      /* area vector */
face_t f;   /* An integer data type that identifies a particular face within
    a face thread : tiny cell face */

Thread *tf;   /* A structure data type that stores data that is common to
   the group of cells or faces that it represents : boundary we define in
   FLUENT (wall, inlet, outlet...) */

 source  = 0.;
 dS[eqn] = 0.;  /* Array that contains the derivative of the source term
     with respect to the dependent variable of the transport equation. */

 c_face_loop(c,t,i)   /* loops over all faces of a given cell (c: cell) */
 {
         f = C_FACE(c,t,i);   /* return the global face index face_t f for
             the given cell_t c, Thread *t, and local face index number i :
             get global face index face_t f */
         tf = C_FACE_THREAD(c,t,i);  /* return the Thread *t of the face_t f
              that is returned by C_FACE  : get Thread *t of the face_t f */

  if(THREAD_ID(C_FACE_THREAD(c,t,i)) == wide_shell_IR[0]) /* if the face is
     on the straight part of the strand */
  {
           F_AREA(A,f,tf);                                          /* F_AREA:
               generates area vector A[ND_ND] as an output */
           A_y = A[1];                                              /*  y
               component of area vector A[2]  */
           source = -C_R(c,t)*vcast* fabs(A_y)* C_U(c,t) /C_VOLUME(c,t);
                     /* C_R(c,t)*vcast* fabs(area_face_cast)*vx/C_VOLUME(c,t
               ) : kg/s^-2/m-2  -> correct unit for momentum */
           dS[eqn] = -C_R(c,t)*vcast* fabs(A_y) /C_VOLUME(c,t);
  }

  for (a = 1; a <= CurveNumber; a++) /* for-loop through all faces on curved
       WF_IR */
  {
        if (THREAD_ID(C_FACE_THREAD(c, t, i)) == wide_shell_IR[a])
        {
              F_AREA(A, f, tf);                 /* F_AREA: generates area
                  vector A[ND_ND] as an output */
              F_CENTROID(x, f, tf);                    /* F_CENTROID:
                  generates coordinate position vector x[ND_ND] as an
                  output */
              x1 = x[0];                        /* x coordinate position of
                  face centroid */
              y1 = x[1];                        /* y coordinate position of
                  face centroid */

              A_N = NV_MAG(A);                                  /* face
                  area magnitude */

       delta_x = fabs(x1-x0[a-1]);
              delta_y = fabs(y1-y_0[a-1]);
              theta = atan((delta_y)/(delta_x));              /* the
                  angle between the horizontal line (x direction) and the
```

```
                    line connecting the face-center point and the acr-center
                    point in radian */

                CA_0 = theta;      /* initialization of the central angle of
                    the arc section the face is in */
                dis = 0;     /* initialization of the distance below the
                    meniscus along the strand */

                if (a == 1)
                {
                        dis = SL + CA_0 * R[0];    /* the distance below the
                            meniscus along the strand in m */
                }
                else
                {
                        for (b = 0; b <= (a-2); b++)
                        {
                                CA_0 -= CA[b];  /* the central angle of the
                                    arc section the face is in */
                        }

                        dis = SL + CA_0 * R[a-1];

                        for (b1 = 0; b1 <= (a-2); b1++)
                        {
                                dis += CA[b1] * R[b1];     /* the distance
                                    below the meniscus along the strand in m
                                    */
                        }

                }

                theta_pro = atan(SC/(2 * sqrt(vcast * dis)));     /* the
                    angle between the face and the casting direction in
                    radian */
                A_pro = A_N * sin(theta_pro);    /* the projected face area
                    in the casting direction in m^2 */

                source = -C_R(c,t)*vcast*A_pro* C_U(c,t) / C_VOLUME(c, t);
                            /* C_R(c,t)*vcast* fabs(area_face_cast)*vx/
                        C_VOLUME(c,t) : kg/s^-2/m-2  -> correct unit for momentum
                         */
                dS[eqn] = -C_R(c,t)*vcast*A_pro / C_VOLUME(c, t);
        }
            }
        }
         return source;
         return dS[eqn];
}

DEFINE_SOURCE(steel_xmom_sink_narrow_shell, c, t, dS, eqn)     /* name, cell, time,
    derivative of the source, eqn number */

{
        real source, x1, y1, delta_x, delta_y, A_y, A_N, A_pro, dis, theta, CA_0,
            theta_pro;
        int i;
        int a;
        int b;
```

```
int b1;
real x[ND_ND];      /* coordinate position vector*/
real A[ND_ND];      /* area vector */
face_t f;   /* An integer data type that identifies a particular face within
    a face thread : tiny cell face */

Thread *tf;   /* A structure data type that stores data that is common to
   the group of cells or faces that it represents : boundary we define in
   FLUENT (wall, inlet, outlet...) */

 source  = 0.;
 dS[eqn] = 0.;  /* Array that contains the derivative of the source term
    with respect to the dependent variable of the transport equation. */

 c_face_loop(c,t,i)   /* loops over all faces of a given cell (c: cell) */
 {
        f = C_FACE(c,t,i);   /* return the global face index face_t f for
            the given cell_t c, Thread *t, and local face index number i :
            get global face index face_t f */
        tf = C_FACE_THREAD(c,t,i);  /* return the Thread *t of the face_t f
             that is returned by C_FACE  : get Thread *t of the face_t f */

  if(THREAD_ID(C_FACE_THREAD(c,t,i)) == narrow_shell[0]) /* if the face is
     on the straight part of the strand */
  {
          F_AREA(A,f,tf);                                          /* F_AREA:
              generates area vector A[ND_ND] as an output */
          A_y = A[1];                                              /*  y
              component of area vector A[2]  */
          source = -C_R(c,t)*vcast* fabs(A_y)* C_U(c,t)/C_VOLUME(c,t);
                    /* C_R(c,t)*vcast* fabs(area_face_cast)*vx/C_VOLUME(c,t
              ) : kg/s^-2/m-2  -> correct unit for momentum */
          dS[eqn] = -C_R(c,t)*vcast* fabs(A_y) /C_VOLUME(c,t);
  }

  for (a = 1; a <= CurveNumber; a++) /* for-loop through all faces on curved
      NF */
  {
        if (THREAD_ID(C_FACE_THREAD(c, t, i)) == narrow_shell[a])
        {
              F_AREA(A, f, tf);                  /* F_AREA: generates area
                  vector A[ND_ND] as an output */
              F_CENTROID(x, f, tf);                      /* F_CENTROID:
                  generates coordinate position vector x[ND_ND] as an
                  output */
              x1 = x[0];                         /* x coordinate position of
                  face centroid */
              y1 = x[1];                         /* y coordinate position of
                  face centroid */

              A_N = NV_MAG(A);                                  /* face
                  area magnitude */

       delta_x = fabs(x1-x0[a-1]);
              delta_y = fabs(y1-y_0[a-1]);
              theta = atan((delta_y)/(delta_x));                /* the
                  angle between the horizontal line (x direction) and the
                  line connecting the face-center point and the acr-center
                  point in radian */
```

```
                        CA_0 = theta;      /* initialization of the central angle of
                            the arc section the face is in */
                        dis = 0;     /* initialization of the distance below the
                            meniscus along the strand */

                        if (a == 1)
                        {
                                dis = SL + CA_0 * R[0];    /* the distance below the
                                    meniscus along the strand in m */
                        }
                        else
                        {
                                for (b = 0; b <= (a-2); b++)
                                {
                                        CA_0 -= CA[b];  /* the central angle of the
                                            arc section the face is in */
                                }

                                dis = SL + CA_0 * R[a-1];

                                for (b1 = 0; b1 <= (a-2); b1++)
                                {
                                        dis += CA[b1] * R[b1];     /* the distance
                                            below the meniscus along the strand in m
                                            */
                                }

                        }

                        theta_pro = atan(SC/(2 * sqrt(vcast * dis)));     /* the
                            angle between the face and the casting direction in
                            radian */
                        A_pro = A_N * sin(theta_pro);   /* the projected face area
                            in the casting direction in m^2 */

                        source = -C_R(c,t)*vcast*A_pro* C_U(c,t) / C_VOLUME(c, t);
                                /* C_R(c,t)*vcast* fabs(area_face_cast)*vx/
                            C_VOLUME(c,t) : kg/s^-2/m-2  -> correct unit for momentum
                             */
                    dS[eqn] = -C_R(c,t)*vcast*A_pro / C_VOLUME(c, t);
            }
                }
            }
             return source;
             return dS[eqn];
}


DEFINE_SOURCE(steel_ymom_sink_wide_shell_OR, c, t, dS, eqn)     /* name, cell, time,
    derivative of the source, eqn number */

{
        real source, x1, y1, delta_x, delta_y, A_y, A_N, A_pro, dis, theta, CA_0,
            theta_pro;
        int i;
        int a;
        int b;
        int b1;
```

```
real x[ND_ND];      /* coordinate position vector*/
real A[ND_ND];      /* area vector */
face_t f;   /* An integer data type that identifies a particular face within
    a face thread : tiny cell face */

Thread *tf;   /* A structure data type that stores data that is common to
   the group of cells or faces that it represents : boundary we define in
   FLUENT (wall, inlet, outlet...) */

 source  = 0.;
 dS[eqn] = 0.;  /* Array that contains the derivative of the source term
     with respect to the dependent variable of the transport equation. */

 c_face_loop(c,t,i)   /* loops over all faces of a given cell (c: cell) */
 {
        f = C_FACE(c,t,i);   /* return the global face index face_t f for
            the given cell_t c, Thread *t, and local face index number i :
            get global face index face_t f */
        tf = C_FACE_THREAD(c,t,i);  /* return the Thread *t of the face_t f
             that is returned by C_FACE  : get Thread *t of the face_t f */

  if(THREAD_ID(C_FACE_THREAD(c,t,i)) == wide_shell_OR[0]) /* if the face is
     on the straight part of the strand */
  {
          F_AREA(A,f,tf);                                          /* F_AREA:
              generates area vector A[ND_ND] as an output */
          A_y = A[1];                                              /*  y
              component of area vector A[2]  */
          source = -C_R(c,t)*vcast* fabs(A_y)* C_V(c,t)/C_VOLUME(c,t);
                     /* C_R(c,t)*vcast* fabs(area_face_y)*vy/C_VOLUME(c,t)
              : kg/s^-2/m-2  -> correct unit for momentum */
          dS[eqn] = -C_R(c,t)*vcast* fabs(A_y) /C_VOLUME(c,t);
  }

  for (a = 1; a <= CurveNumber; a++) /* for-loop through all faces on curved
      WF_OR */
  {
        if (THREAD_ID(C_FACE_THREAD(c, t, i)) == wide_shell_OR[a])
        {
              F_AREA(A, f, tf);                  /* F_AREA: generates area
                  vector A[ND_ND] as an output */
              F_CENTROID(x, f, tf);                      /* F_CENTROID:
                  generates coordinate position vector x[ND_ND] as an
                  output */
              x1 = x[0];                         /* x coordinate position of
                  face centroid */
              y1 = x[1];                         /* y coordinate position of
                  face centroid */

              A_N = NV_MAG(A);                                      /* face
                  area magnitude */

       delta_x = fabs(x1-x0[a-1]);
              delta_y = fabs(y1-y_0[a-1]);
              theta = atan((delta_y)/(delta_x));                    /* the
                  angle between the horizontal line (x direction) and the
                  line connecting the face-center point and the acr-center
                  point in radian */
```

```
                        CA_0 = theta;      /* initialization of the central angle of
                            the arc section the face is in */
                        dis = 0;     /* initialization of the distance below the
                            meniscus along the strand */

                        if (a == 1)
                        {
                                dis = SL + CA_0 * R[0];    /* the distance below the
                                    meniscus along the strand in m */
                        }
                        else
                        {
                                for (b = 0; b <= (a-2); b++)
                                {
                                        CA_0 -= CA[b];  /* the central angle of the
                                            arc section the face is in */
                                }

                                dis = SL + CA_0 * R[a-1];

                                for (b1 = 0; b1 <= (a-2); b1++)
                                {
                                        dis += CA[b1] * R[b1];     /* the distance
                                            below the meniscus along the strand in m
                                            */
                                }

                        }

                        theta_pro = atan(SC/(2 * sqrt(vcast * dis)));     /* the
                            angle between the face and the casting direction in
                            radian */
                        A_pro = A_N * sin(theta_pro);    /* the projected face area
                            in the casting direction in m^2 */

                        source = -C_R(c,t)*vcast*A_pro* C_V(c,t) / C_VOLUME(c, t);
                                /* C_R(c,t)*vcast* fabs(area_face_cast)*vy/
                            C_VOLUME(c,t) : kg/s^-2/m-2  -> correct unit for momentum
                             */
                    dS[eqn] = -C_R(c,t)*vcast*A_pro / C_VOLUME(c, t);
                }
                        }
                }
                 return source;
                 return dS[eqn];
}

DEFINE_SOURCE(steel_ymom_sink_wide_shell_IR, c, t, dS, eqn)     /* name, cell, time,
    derivative of the source, eqn number */

{
        real source, x1, y1, delta_x, delta_y, A_y, A_N, A_pro, dis, theta, CA_0,
            theta_pro;
        int i;
        int a;
        int b;
        int b1;
        real x[ND_ND];      /* coordinate position vector*/
        real A[ND_ND];      /* area vector */
```

```
face_t f;   /* An integer data type that identifies a particular face within
    a face thread : tiny cell face */

Thread *tf;   /* A structure data type that stores data that is common to
   the group of cells or faces that it represents : boundary we define in
   FLUENT (wall, inlet, outlet...) */

source  = 0.;
dS[eqn] = 0.;  /* Array that contains the derivative of the source term
    with respect to the dependent variable of the transport equation. */

c_face_loop(c,t,i)   /* loops over all faces of a given cell (c: cell) */
{
        f = C_FACE(c,t,i);   /* return the global face index face_t f for
            the given cell_t c, Thread *t, and local face index number i :
            get global face index face_t f */
        tf = C_FACE_THREAD(c,t,i);  /* return the Thread *t of the face_t f
             that is returned by C_FACE  : get Thread *t of the face_t f */

 if(THREAD_ID(C_FACE_THREAD(c,t,i)) == wide_shell_IR[0]) /* if the face is
    on the straight part of the strand */
 {
         F_AREA(A,f,tf);                                          /* F_AREA:
             generates area vector A[ND_ND] as an output */
         A_y = A[1];                                              /*  y
             component of area vector A[2]  */
         source = -C_R(c,t)*vcast* fabs(A_y)* C_V(c,t)/C_VOLUME(c,t);
                   /* C_R(c,t)*vcast* fabs(area_face_y)*vy/C_VOLUME(c,t) :
              kg/s^-2/m-2  -> correct unit for momentum */
         dS[eqn] = -C_R(c,t)*vcast* fabs(A_y) /C_VOLUME(c,t);
 }

 for (a = 1; a <= CurveNumber; a++) /* for-loop through all faces on curved
     WF_IR */
 {
       if (THREAD_ID(C_FACE_THREAD(c, t, i)) == wide_shell_IR[a])
       {
             F_AREA(A, f, tf);              /* F_AREA: generates area
                 vector A[ND_ND] as an output */
             F_CENTROID(x, f, tf);                  /* F_CENTROID:
                 generates coordinate position vector x[ND_ND] as an
                 output */
             x1 = x[0];                     /* x coordinate position of
                 face centroid */
             y1 = x[1];                     /* y coordinate position of
                 face centroid */

             A_N = NV_MAG(A);                                  /* face
                 area magnitude */

      delta_x = fabs(x1-x0[a-1]);
             delta_y = fabs(y1-y_0[a-1]);
             theta = atan((delta_y)/(delta_x));                /* the
                 angle between the horizontal line (x direction) and the
                 line connecting the face-center point and the acr-center
                 point in radian */

             CA_0 = theta;      /* initialization of the central angle of
                 the arc section the face is in */
```

```
                        dis = 0;    /* initialization of the distance below the
                            meniscus along the strand */

                        if (a == 1)
                        {
                                dis = SL + CA_0 * R[0];   /* the distance below the
                                    meniscus along the strand in m */
                        }
                        else
                        {
                                for (b = 0; b <= (a-2); b++)
                                {
                                        CA_0 -= CA[b];  /* the central angle of the
                                            arc section the face is in */
                                }

                                dis = SL + CA_0 * R[a-1];

                                for (b1 = 0; b1 <= (a-2); b1++)
                                {
                                        dis += CA[b1] * R[b1];    /* the distance
                                            below the meniscus along the strand in m
                                            */
                                }

                        }

                        theta_pro = atan(SC/(2 * sqrt(vcast * dis)));    /* the
                            angle between the face and the casting direction in
                            radian */
                        A_pro = A_N * sin(theta_pro);   /* the projected face area
                            in the casting direction in m^2 */

                        source = -C_R(c,t)*vcast*A_pro* C_V(c,t) / C_VOLUME(c, t);
                                /* C_R(c,t)*vcast* fabs(area_face_cast)*vy/
                            C_VOLUME(c,t) : kg/s^-2/m-2 -> correct unit for momentum
                             */
                    dS[eqn] = -C_R(c,t)*vcast*A_pro / C_VOLUME(c, t);
            }
                }
            }
             return source;
             return dS[eqn];
}

DEFINE_SOURCE(steel_ymom_sink_narrow_shell, c, t, dS, eqn)    /* name, cell, time,
    derivative of the source, eqn number */

{
        real source, x1, y1, delta_x, delta_y, A_y, A_N, A_pro, dis, theta, CA_0,
            theta_pro;
        int i;
        int a;
        int b;
        int b1;
        real x[ND_ND];      /* coordinate position vector*/
        real A[ND_ND];      /* area vector */
        face_t f;   /* An integer data type that identifies a particular face within
             a face thread : tiny cell face */
```

```
Thread *tf;   /* A structure data type that stores data that is common to
   the group of cells or faces that it represents : boundary we define in
   FLUENT (wall, inlet, outlet...) */

source  = 0.;
dS[eqn] = 0.;  /* Array that contains the derivative of the source term
    with respect to the dependent variable of the transport equation. */

c_face_loop(c,t,i)   /* loops over all faces of a given cell (c: cell) */
{
        f = C_FACE(c,t,i);   /* return the global face index face_t f for
            the given cell_t c, Thread *t, and local face index number i :
            get global face index face_t f */
        tf = C_FACE_THREAD(c,t,i);  /* return the Thread *t of the face_t f
             that is returned by C_FACE  : get Thread *t of the face_t f */

 if(THREAD_ID(C_FACE_THREAD(c,t,i)) == narrow_shell[0]) /* if the face is
    on the straight part of the strand */
  {
          F_AREA(A,f,tf);                                          /* F_AREA:
              generates area vector A[ND_ND] as an output */
          A_y = A[1];                                              /*  y
              component of area vector A[2]  */
          source = -C_R(c,t)*vcast* fabs(A_y)* C_V(c,t)/C_VOLUME(c,t);
                    /* C_R(c,t)*vcast* fabs(area_face_y)*vy/C_VOLUME(c,t) :
               kg/s^-2/m-2  -> correct unit for momentum */
          dS[eqn] = -C_R(c,t)*vcast* fabs(A_y) /C_VOLUME(c,t);
  }

 for (a = 1; a <= CurveNumber; a++) /* for-loop through all faces on curved
     NF */
  {
        if (THREAD_ID(C_FACE_THREAD(c, t, i)) == narrow_shell[a])
        {
              F_AREA(A, f, tf);              /* F_AREA: generates area
                  vector A[ND_ND] as an output */
              F_CENTROID(x, f, tf);                  /* F_CENTROID:
                  generates coordinate position vector x[ND_ND] as an
                  output */
              x1 = x[0];                     /* x coordinate position of
                  face centroid */
              y1 = x[1];                     /* y coordinate position of
                  face centroid */

              A_N = NV_MAG(A);                                  /* face
                  area magnitude */

      delta_x = fabs(x1-x0[a-1]);
              delta_y = fabs(y1-y_0[a-1]);
              theta = atan((delta_y)/(delta_x));              /* the
                  angle between the horizontal line (x direction) and the
                  line connecting the face-center point and the acr-center
                  point in radian */

              CA_0 = theta;      /* initialization of the central angle of
                  the arc section the face is in */
              dis = 0;     /* initialization of the distance below the
                  meniscus along the strand */
```

```
                        if (a == 1)
                        {
                                dis = SL + CA_0 * R[0];    /* the distance below the
                                    meniscus along the strand in m */
                        }
                        else
                        {
                                for (b = 0; b <= (a-2); b++)
                                {
                                        CA_0 -= CA[b];  /* the central angle of the
                                            arc section the face is in */
                                }

                                dis = SL + CA_0 * R[a-1];

                                for (b1 = 0; b1 <= (a-2); b1++)
                                {
                                        dis += CA[b1] * R[b1];     /* the distance
                                            below the meniscus along the strand in m
                                            */
                                }

                        }

                        theta_pro = atan(SC/(2 * sqrt(vcast * dis)));     /* the
                            angle between the face and the casting direction in
                            radian */
                        A_pro = A_N * sin(theta_pro);   /* the projected face area
                            in the casting direction in m^2 */

                        source = -C_R(c,t)*vcast*A_pro* C_V(c,t) / C_VOLUME(c, t);
                                /* C_R(c,t)*vcast* fabs(area_face_cast)*vy/
                            C_VOLUME(c,t) : kg/s^-2/m-2 -> correct unit for momentum
                             */
                    dS[eqn] = -C_R(c,t)*vcast*A_pro / C_VOLUME(c, t);
            }
                }
            }
             return source;
             return dS[eqn];
}



DEFINE_SOURCE(steel_zmom_sink_wide_shell_OR, c, t, dS, eqn)     /* name, cell, time,
    derivative of the source, eqn number */

{
        real source, x1, y1, delta_x, delta_y, A_y, A_N, A_pro, dis, theta, CA_0,
            theta_pro;
        int i;
        int a;
        int b;
        int b1;
        real x[ND_ND];      /* coordinate position vector*/
        real A[ND_ND];      /* area vector */
        face_t f;   /* An integer data type that identifies a particular face within
             a face thread : tiny cell face */
```

```
Thread *tf;   /* A structure data type that stores data that is common to
   the group of cells or faces that it represents : boundary we define in
   FLUENT (wall, inlet, outlet...) */

source  = 0.;
dS[eqn] = 0.;  /* Array that contains the derivative of the source term
    with respect to the dependent variable of the transport equation. */

c_face_loop(c,t,i)   /* loops over all faces of a given cell (c: cell) */
{
        f = C_FACE(c,t,i);   /* return the global face index face_t f for
            the given cell_t c, Thread *t, and local face index number i :
            get global face index face_t f */
        tf = C_FACE_THREAD(c,t,i);  /* return the Thread *t of the face_t f
             that is returned by C_FACE  : get Thread *t of the face_t f */

 if(THREAD_ID(C_FACE_THREAD(c,t,i)) == wide_shell_OR[0]) /* if the face is
    on the straight part of the strand */
  {
          F_AREA(A,f,tf);                                          /* F_AREA:
              generates area vector A[ND_ND] as an output */
          A_y = A[1];                                              /*  y
              component of area vector A[2]  */
          source = -C_R(c,t)*vcast* fabs(A_y)* C_W(c,t)/C_VOLUME(c,t);
                    /* C_R(c,t)*vcast* fabs(area_face_y)*vz/C_VOLUME(c,t) :
               kg/s^-2/m-2  -> correct unit for momentum */
          dS[eqn] = -C_R(c,t)*vcast* fabs(A_y) /C_VOLUME(c,t);
  }

 for (a = 1; a <= CurveNumber; a++) /* for-loop through all faces on curved
     WF_OR */
  {
        if (THREAD_ID(C_FACE_THREAD(c, t, i)) == wide_shell_OR[a])
        {
                F_AREA(A, f, tf);                 /* F_AREA: generates area
                    vector A[ND_ND] as an output */
                F_CENTROID(x, f, tf);                    /* F_CENTROID:
                    generates coordinate position vector x[ND_ND] as an
                    output */
                x1 = x[0];                        /* x coordinate position of
                    face centroid */
                y1 = x[1];                        /* y coordinate position of
                    face centroid */

                A_N = NV_MAG(A);                                  /* face
                    area magnitude */

      delta_x = fabs(x1-x0[a-1]);
                delta_y = fabs(y1-y_0[a-1]);
                theta = atan((delta_y)/(delta_x));                /* the
                    angle between the horizontal line (x direction) and the
                    line connecting the face-center point and the acr-center
                    point in radian */

                CA_0 = theta;      /* initialization of the central angle of
                    the arc section the face is in */
                dis = 0;     /* initialization of the distance below the
                    meniscus along the strand */
```

```
                        if (a == 1)
                        {
                                dis = SL + CA_0 * R[0];    /* the distance below the
                                    meniscus along the strand in m */
                        }
                        else
                        {
                                for (b = 0; b <= (a-2); b++)
                                {
                                        CA_0 -= CA[b];  /* the central angle of the
                                            arc section the face is in */
                                }

                                dis = SL + CA_0 * R[a-1];

                                for (b1 = 0; b1 <= (a-2); b1++)
                                {
                                        dis += CA[b1] * R[b1];     /* the distance
                                            below the meniscus along the strand in m
                                            */
                                }

                        }

                        theta_pro = atan(SC/(2 * sqrt(vcast * dis)));     /* the
                            angle between the face and the casting direction in
                            radian */
                        A_pro = A_N * sin(theta_pro);   /* the projected face area
                            in the casting direction in m^2 */

                        source = -C_R(c,t)*vcast*A_pro* C_W(c,t) / C_VOLUME(c, t);
                                    /* C_R(c,t)*vcast* fabs(area_face_cast)*vz/
                           C_VOLUME(c,t) : kg/s^-2/m-2 -> correct unit for momentum
                            */
                    dS[eqn] = -C_R(c,t)*vcast*A_pro / C_VOLUME(c, t);
            }
                }
            }
             return source;
             return dS[eqn];
}

DEFINE_SOURCE(steel_zmom_sink_wide_shell_IR, c, t, dS, eqn)     /* name, cell, time,
    derivative of the source, eqn number */

{
        real source, x1, y1, delta_x, delta_y, A_y, A_N, A_pro, dis, theta, CA_0,
            theta_pro;
        int i;
        int a;
        int b;
        int b1;
        real x[ND_ND];      /* coordinate position vector*/
        real A[ND_ND];      /* area vector */
        face_t f;   /* An integer data type that identifies a particular face within
             a face thread : tiny cell face */
```

```
Thread *tf;   /* A structure data type that stores data that is common to
   the group of cells or faces that it represents : boundary we define in
   FLUENT (wall, inlet, outlet...) */

source  = 0.;
dS[eqn] = 0.;  /* Array that contains the derivative of the source term
    with respect to the dependent variable of the transport equation. */

c_face_loop(c,t,i)   /* loops over all faces of a given cell (c: cell) */
{
        f = C_FACE(c,t,i);   /* return the global face index face_t f for
            the given cell_t c, Thread *t, and local face index number i :
            get global face index face_t f */
        tf = C_FACE_THREAD(c,t,i);  /* return the Thread *t of the face_t f
             that is returned by C_FACE  : get Thread *t of the face_t f */

 if(THREAD_ID(C_FACE_THREAD(c,t,i)) == wide_shell_IR[0]) /* if the face is
    on the straight part of the strand */
  {
          F_AREA(A,f,tf);                                          /* F_AREA:
              generates area vector A[ND_ND] as an output */
          A_y = A[1];                                              /*  y
              component of area vector A[2]  */
          source = -C_R(c,t)*vcast* fabs(A_y)* C_W(c,t)/C_VOLUME(c,t);
                    /* C_R(c,t)*vcast* fabs(area_face_cast)*vz/C_VOLUME(c,t)
           : kg/s^-2/m-2  -> correct unit for momentum */
          dS[eqn] = -C_R(c,t)*vcast* fabs(A_y) /C_VOLUME(c,t);
  }

 for (a = 1; a <= CurveNumber; a++) /* for-loop through all faces on curved
     WF_IR */
  {
        if (THREAD_ID(C_FACE_THREAD(c, t, i)) == wide_shell_IR[a])
        {
              F_AREA(A, f, tf);                  /* F_AREA: generates area
                  vector A[ND_ND] as an output */
              F_CENTROID(x, f, tf);                      /* F_CENTROID:
                  generates coordinate position vector x[ND_ND] as an
                  output */
              x1 = x[0];                         /* x coordinate position of
                  face centroid */
              y1 = x[1];                         /* y coordinate position of
                  face centroid */

              A_N = NV_MAG(A);                                  /* face
                  area magnitude */

      delta_x = fabs(x1-x0[a-1]);
              delta_y = fabs(y1-y_0[a-1]);
              theta = atan((delta_y)/(delta_x));                /* the
                  angle between the horizontal line (x direction) and the
                  line connecting the face-center point and the acr-center
                  point in radian */

              CA_0 = theta;      /* initialization of the central angle of
                  the arc section the face is in */
              dis = 0;     /* initialization of the distance below the
                  meniscus along the strand */
```

```
                        if (a == 1)
                        {
                                dis = SL + CA_0 * R[0];    /* the distance below the
                                    meniscus along the strand in m */
                        }
                        else
                        {
                                for (b = 0; b <= (a-2); b++)
                                {
                                        CA_0 -= CA[b];  /* the central angle of the
                                            arc section the face is in */
                                }

                                dis = SL + CA_0 * R[a-1];

                                for (b1 = 0; b1 <= (a-2); b1++)
                                {
                                        dis += CA[b1] * R[b1];     /* the distance
                                            below the meniscus along the strand in m
                                            */
                                }

                        }

                        theta_pro = atan(SC/(2 * sqrt(vcast * dis)));     /* the
                            angle between the face and the casting direction in
                            radian */
                        A_pro = A_N * sin(theta_pro);   /* the projected face area
                            in the casting direction in m^2 */

                        source = -C_R(c,t)*vcast*A_pro* C_W(c,t) / C_VOLUME(c, t);
                                /* C_R(c,t)*vcast* fabs(area_face_cast)*vz/
                            C_VOLUME(c,t) : kg/s^-2/m-2 -> correct unit for
                            continuity */
                    dS[eqn] = -C_R(c,t)*vcast*A_pro / C_VOLUME(c, t);
            }
                }
            }
             return source;
             return dS[eqn];
}

DEFINE_SOURCE(steel_zmom_sink_narrow_shell, c, t, dS, eqn)     /* name, cell, time,
    derivative of the source, eqn number */

{
        real source, x1, y1, delta_x, delta_y, A_y, A_N, A_pro, dis, theta, CA_0,
            theta_pro;
        int i;
        int a;
        int b;
        int b1;
        real x[ND_ND];      /* coordinate position vector*/
        real A[ND_ND];      /* area vector */
        face_t f;   /* An integer data type that identifies a particular face within
             a face thread : tiny cell face */

        Thread *tf;   /* A structure data type that stores data that is common to
            the group of cells or faces that it represents : boundary we define in
```

```
  FLUENT (wall, inlet, outlet...) */

source  = 0.;
dS[eqn] = 0.;  /* Array that contains the derivative of the source term
    with respect to the dependent variable of the transport equation. */

c_face_loop(c,t,i)   /* loops over all faces of a given cell (c: cell) */
{
        f = C_FACE(c,t,i);   /* return the global face index face_t f for
            the given cell_t c, Thread *t, and local face index number i :
            get global face index face_t f */
        tf = C_FACE_THREAD(c,t,i);  /* return the Thread *t of the face_t f
             that is returned by C_FACE  : get Thread *t of the face_t f */

 if(THREAD_ID(C_FACE_THREAD(c,t,i)) == narrow_shell[0]) /* if the face is
    on the straight part of the strand */
 {
          F_AREA(A,f,tf);                                          /* F_AREA:
              generates area vector A[ND_ND] as an output */
          A_y = A[1];                                              /*  y
              component of area vector A[2]  */
          source = -C_R(c,t)*vcast* fabs(A_y)* C_W(c,t)/C_VOLUME(c,t);
                    /* C_R(c,t)*vcast* fabs(area_face_cast)*vz/C_VOLUME(c,t
              ) : kg/s^-2/m-2  -> correct unit for momentum */
          dS[eqn] = -C_R(c,t)*vcast* fabs(A_y) /C_VOLUME(c,t);
 }

 for (a = 1; a <= CurveNumber; a++) /* for-loop through all faces on curved
     NF */
 {
        if (THREAD_ID(C_FACE_THREAD(c, t, i)) == narrow_shell[a])
        {
              F_AREA(A, f, tf);                 /* F_AREA: generates area
                  vector A[ND_ND] as an output */
              F_CENTROID(x, f, tf);                    /* F_CENTROID:
                  generates coordinate position vector x[ND_ND] as an
                  output */
              x1 = x[0];                        /* x coordinate position of
                  face centroid */
              y1 = x[1];                        /* y coordinate position of
                  face centroid */

              A_N = NV_MAG(A);                                  /* face
                  area magnitude */

      delta_x = fabs(x1-x0[a-1]);
              delta_y = fabs(y1-y_0[a-1]);
              theta = atan((delta_y)/(delta_x));                /* the
                  angle between the horizontal line (x direction) and the
                  line connecting the face-center point and the acr-center
                  point in radian */

              CA_0 = theta;      /* initialization of the central angle of
                  the arc section the face is in */
              dis = 0;     /* initialization of the distance below the
                  meniscus along the strand */

              if (a == 1)
              {
```

```
                        dis = SL + CA_0 * R[0];    /* the distance below the
                            meniscus along the strand in m */
                    }
                    else
                    {
                        for (b = 0; b <= (a-2); b++)
                        {
                            CA_0 -= CA[b];  /* the central angle of the
                                arc section the face is in */
                        }

                        dis = SL + CA_0 * R[a-1];

                        for (b1 = 0; b1 <= (a-2); b1++)
                        {
                            dis += CA[b1] * R[b1];     /* the distance
                                below the meniscus along the strand in m
                                */
                        }

                    }

                    theta_pro = atan(SC/(2 * sqrt(vcast * dis)));     /* the
                        angle between the face and the casting direction in
                        radian */
                    A_pro = A_N * sin(theta_pro);    /* the projected face area
                        in the casting direction in m^2 */

                    source = -C_R(c,t)*vcast*A_pro* C_W(c,t) / C_VOLUME(c, t);
                            /* C_R(c,t)*vcast* fabs(area_face_cast)*vz/
                        C_VOLUME(c,t) : kg/s^-2/m-2  -> correct unit for momentum
                         */
                dS[eqn] = -C_R(c,t)*vcast*A_pro / C_VOLUME(c, t);
            }
        }
        }
         return source;
         return dS[eqn];
}

/*----------------------------------------------------------Energy Sink
    ----------------------------------------------------------*/
DEFINE_SOURCE(steel_energy_sink_wide_shell_OR, c, t, dS, eqn)     /* name, cell, time
    , derivative of the source, eqn number */

{
        real source, x1, y1, delta_x, delta_y, A_y, A_N, A_pro, dis, theta, CA_0,
            theta_pro;
        int i;
        int a;
        int b;
        int b1;
        real x[ND_ND];      /* coordinate position vector*/
        real A[ND_ND];      /* area vector */
        face_t f;   /* An integer data type that identifies a particular face within
             a face thread : tiny cell face */

        Thread *tf;   /* A structure data type that stores data that is common to
            the group of cells or faces that it represents : boundary we define in
```

```
   FLUENT (wall, inlet, outlet...) */

source  = 0.;
dS[eqn] = 0.;  /* Array that contains the derivative of the source term
    with respect to the dependent variable of the transport equation. */

c_face_loop(c,t,i)   /* loops over all faces of a given cell (c: cell) */
{
        f = C_FACE(c,t,i);   /* return the global face index face_t f for
            the given cell_t c, Thread *t, and local face index number i :
            get global face index face_t f */
        tf = C_FACE_THREAD(c,t,i);  /* return the Thread *t of the face_t f
             that is returned by C_FACE  : get Thread *t of the face_t f */

 if(THREAD_ID(C_FACE_THREAD(c,t,i)) == wide_shell_OR[0]) /* if the face is
    on the straight part of the strand */
 {
          F_AREA(A,f,tf);                                          /* F_AREA:
              generates area vector A[ND_ND] as an output */
          A_y = A[1];                                              /*  y
              component of area vector A[2]  */
          source = -C_R(c,t)*vcast* fabs(A_y)* cp * (C_T(c,t) - Tref)/
              C_VOLUME(c,t);
          dS[eqn] = -C_R(c,t)*vcast*fabs(A_y)* cp / C_VOLUME(c,t) + 0.7 *
              vcast* fabs(A_y)* cp *(C_T(c,t) - Tref) / C_VOLUME(c,t);
 }

 for (a = 1; a <= CurveNumber; a++) /* for-loop through all faces on curved
     WF_OR */
 {
       if (THREAD_ID(C_FACE_THREAD(c, t, i)) == wide_shell_OR[a])
       {
             F_AREA(A, f, tf);                  /* F_AREA: generates area
                 vector A[ND_ND] as an output */
             F_CENTROID(x, f, tf);                    /* F_CENTROID:
                 generates coordinate position vector x[ND_ND] as an
                 output */
             x1 = x[0];                         /* x coordinate position of
                 face centroid */
             y1 = x[1];                         /* y coordinate position of
                 face centroid */

             A_N = NV_MAG(A);                                  /* face
                 area magnitude */

      delta_x = fabs(x1-x0[a-1]);
             delta_y = fabs(y1-y_0[a-1]);
             theta = atan((delta_y)/(delta_x));               /* the
                 angle between the horizontal line (x direction) and the
                 line connecting the face-center point and the acr-center
                 point in radian */

             CA_0 = theta;      /* initialization of the central angle of
                 the arc section the face is in */
             dis = 0;     /* initialization of the distance below the
                 meniscus along the strand */

             if (a == 1)
             {
```

```
                        dis = SL + CA_0 * R[0];   /* the distance below the
                            meniscus along the strand in m */
                    }
                    else
                    {
                        for (b = 0; b <= (a-2); b++)
                        {
                            CA_0 -= CA[b];  /* the central angle of the
                                arc section the face is in */
                        }

                        dis = SL + CA_0 * R[a-1];

                        for (b1 = 0; b1 <= (a-2); b1++)
                        {
                            dis += CA[b1] * R[b1];    /* the distance
                                below the meniscus along the strand in m
                                */
                        }

                    }

                    theta_pro = atan(SC/(2 * sqrt(vcast * dis)));    /* the
                        angle between the face and the casting direction in
                        radian */
                    A_pro = A_N * sin(theta_pro);                    /* the
                        projected face area in the casting direction in m^2 */

                    source = -C_R(c,t)*vcast*A_pro * cp * (C_T(c,t) - Tref) /
                        C_VOLUME(c, t);
                dS[eqn] = -C_R(c,t)*vcast*A_pro* cp / C_VOLUME(c,t) + 0.7*vcast*
                    A_pro* cp * (C_T(c,t) - Tref) / C_VOLUME(c,t);
            }
        }
        }
         return source;
         return dS[eqn];
}

DEFINE_SOURCE(steel_energy_sink_wide_shell_IR, c, t, dS, eqn)    /* name, cell, time
    , derivative of the source, eqn number */

{
        real source, x1, y1, delta_x, delta_y, A_y, A_N, A_pro, dis, theta, CA_0,
            theta_pro;
        int i;
        int a;
        int b;
        int b1;
        real x[ND_ND];      /* coordinate position vector*/
        real A[ND_ND];      /* area vector */
        face_t f;   /* An integer data type that identifies a particular face within
            a face thread : tiny cell face */

        Thread *tf;   /* A structure data type that stores data that is common to
            the group of cells or faces that it represents : boundary we define in
            FLUENT (wall, inlet, outlet...) */

         source  = 0.;
```

```
dS[eqn] = 0.;  /* Array that contains the derivative of the source term
    with respect to the dependent variable of the transport equation. */

c_face_loop(c,t,i)   /* loops over all faces of a given cell (c: cell) */
{
        f = C_FACE(c,t,i);   /* return the global face index face_t f for
            the given cell_t c, Thread *t, and local face index number i :
            get global face index face_t f */
        tf = C_FACE_THREAD(c,t,i);  /* return the Thread *t of the face_t f
             that is returned by C_FACE  : get Thread *t of the face_t f */

 if(THREAD_ID(C_FACE_THREAD(c,t,i)) == wide_shell_IR[0]) /* if the face is
    on the straight part of the strand */
 {
          F_AREA(A,f,tf);                                          /* F_AREA:
              generates area vector A[ND_ND] as an output */
          A_y = A[1];                                              /*  y
              component of area vector A[2]  */
          source = -C_R(c,t)*vcast* fabs(A_y)* cp * (C_T(c,t) - Tref)/
              C_VOLUME(c,t);          /* C_R(c,t)*vcast* fabs(area_face)/
              C_VOLUME(c,t) : kg/s/m-3 -> correct unit for continuity */
          dS[eqn] = -C_R(c,t)*vcast*fabs(A_y)* cp / C_VOLUME(c,t) + 0.7 *
              vcast* fabs(A_y)* cp *(C_T(c,t) - Tref) / C_VOLUME(c,t);
 }

 for (a = 1; a <= CurveNumber; a++) /* for-loop through all faces on curved
     WF_IR */
 {
        if (THREAD_ID(C_FACE_THREAD(c, t, i)) == wide_shell_IR[a])
        {
              F_AREA(A, f, tf);                 /* F_AREA: generates area
                  vector A[ND_ND] as an output */
              F_CENTROID(x, f, tf);                      /* F_CENTROID:
                  generates coordinate position vector x[ND_ND] as an
                  output */
              x1 = x[0];                        /* x coordinate position of
                  face centroid */
              y1 = x[1];                        /* y coordinate position of
                  face centroid */

              A_N = NV_MAG(A);                                  /* face
                  area magnitude */

      delta_x = fabs(x1-x0[a-1]);
              delta_y = fabs(y1-y_0[a-1]);
              theta = atan((delta_y)/(delta_x));                /* the
                  angle between the horizontal line (x direction) and the
                  line connecting the face-center point and the acr-center
                  point in radian */

              CA_0 = theta;      /* initialization of the central angle of
                  the arc section the face is in */
              dis = 0;     /* initialization of the distance below the
                  meniscus along the strand */

              if (a == 1)
              {
                    dis = SL + CA_0 * R[0];    /* the distance below the
                        meniscus along the strand in m */
```

```
                        }
                        else
                        {
                                for (b = 0; b <= (a-2); b++)
                                {
                                        CA_0 -= CA[b];  /* the central angle of the
                                            arc section the face is in */
                                }

                                dis = SL + CA_0 * R[a-1];

                                for (b1 = 0; b1 <= (a-2); b1++)
                                {
                                        dis += CA[b1] * R[b1];     /* the distance
                                            below the meniscus along the strand in m
                                            */
                                }

                        }

                        theta_pro = atan(SC/(2 * sqrt(vcast * dis)));     /* the
                            angle between the face and the casting direction in
                            radian */
                        A_pro = A_N * sin(theta_pro);   /* the projected face area
                            in the casting direction in m^2 */

                        source = -C_R(c,t)*vcast*A_pro * cp * (C_T(c,t) - Tref)/
                            C_VOLUME(c,t);;
                    dS[eqn] = -C_R(c,t)*vcast*A_pro* cp / C_VOLUME(c,t) + 0.7*vcast*
                        A_pro* cp * (C_T(c,t) - Tref) / C_VOLUME(c,t);
            }
                }
            }
             return source;
             return dS[eqn];
}

DEFINE_SOURCE(steel_energy_sink_narrow_shell, c, t, dS, eqn)     /* name, cell, time,
     derivative of the source, eqn number */

{
        real source, x1, y1, delta_x, delta_y, A_y, A_N, A_pro, dis, theta, CA_0,
            theta_pro;
        int i;
        int a;
        int b;
        int b1;
        real x[ND_ND];      /* coordinate position vector*/
        real A[ND_ND];      /* area vector */
        face_t f;   /* An integer data type that identifies a particular face within
             a face thread : tiny cell face */

        Thread *tf;   /* A structure data type that stores data that is common to
            the group of cells or faces that it represents : boundary we define in
            FLUENT (wall, inlet, outlet...) */

         source  = 0.;
         dS[eqn] = 0.;  /* Array that contains the derivative of the source term
             with respect to the dependent variable of the transport equation. */
```

```
c_face_loop(c,t,i)   /* loops over all faces of a given cell (c: cell) */
{
        f = C_FACE(c,t,i);   /* return the global face index face_t f for
            the given cell_t c, Thread *t, and local face index number i :
            get global face index face_t f */
        tf = C_FACE_THREAD(c,t,i);  /* return the Thread *t of the face_t f
             that is returned by C_FACE  : get Thread *t of the face_t f */

 if(THREAD_ID(C_FACE_THREAD(c,t,i)) == narrow_shell[0]) /* if the face is
    on the straight part of the strand */
 {
          F_AREA(A,f,tf);                                          /* F_AREA:
              generates area vector A[ND_ND] as an output */
          A_y = A[1];                                              /*  y
              component of area vector A[2]  */
          source = -C_R(c,t)*vcast* fabs(A_y) * cp * (C_T(c,t) - Tref)/
              C_VOLUME(c,t);          /* C_R(c,t)*vcast* fabs(area_face)/
              C_VOLUME(c,t) : kg/s/m-3 -> correct unit for continuity */
          dS[eqn] = -C_R(c,t)*vcast*fabs(A_y)* cp / C_VOLUME(c,t) + 0.7 *
              vcast* fabs(A_y)* cp *(C_T(c,t) - Tref) / C_VOLUME(c,t);
 }

 for (a = 1; a <= CurveNumber; a++) /* for-loop through all faces on curved
     NF */
 {
        if (THREAD_ID(C_FACE_THREAD(c, t, i)) == narrow_shell[a])
        {
                F_AREA(A, f, tf);                  /* F_AREA: generates area
                    vector A[ND_ND] as an output */
                F_CENTROID(x, f, tf);                       /* F_CENTROID:
                    generates coordinate position vector x[ND_ND] as an
                    output */
                x1 = x[0];                         /* x coordinate position of
                    face centroid */
                y1 = x[1];                         /* y coordinate position of
                    face centroid */

                A_N = NV_MAG(A);                                  /* face
                    area magnitude */

       delta_x = fabs(x1-x0[a-1]);
                delta_y = fabs(y1-y_0[a-1]);
                theta = atan((delta_y)/(delta_x));                /* the
                    angle between the horizontal line (x direction) and the
                    line connecting the face-center point and the acr-center
                    point in radian */

                CA_0 = theta;      /* initialization of the central angle of
                    the arc section the face is in */
                dis = 0;     /* initialization of the distance below the
                    meniscus along the strand */

                if (a == 1)
                {
                        dis = SL + CA_0 * R[0];    /* the distance below the
                            meniscus along the strand in m */
                }
                else
```

```
                        {
                                for (b = 0; b <= (a-2); b++)
                                {
                                        CA_0 -= CA[b];  /* the central angle of the
                                            arc section the face is in */
                                }

                                dis = SL + CA_0 * R[a-1];

                                for (b1 = 0; b1 <= (a-2); b1++)
                                {
                                        dis += CA[b1] * R[b1];     /* the distance
                                            below the meniscus along the strand in m
                                            */
                                }

                        }

                        theta_pro = atan(SC/(2 * sqrt(vcast * dis)));     /* the
                            angle between the face and the casting direction in
                            radian */
                        A_pro = A_N * sin(theta_pro);   /* the projected face area
                            in the casting direction in m^2 */

                        source = -C_R(c,t)*vcast*A_pro * cp * (C_T(c,t) - Tref) /
                            C_VOLUME(c, t);
                    dS[eqn] = -C_R(c,t)*vcast*A_pro* cp / C_VOLUME(c,t) + 0.7*vcast*
                        A_pro* cp * (C_T(c,t) - Tref) / C_VOLUME(c,t);
            }
                }
            }
             return source;
             return dS[eqn];
}
```

Listing A.4: Hook Capture Mechanism on a Haf-Domain Strand

```
/************************************************************/
/* Hook Capture */
/************************************************************/

/* Disable the warning c4996 from compiler using fscanf and fopen */
#pragma warning(disable : 4996)

#include "udf.h"
#include "mem.h"
#include "sg.h"
#include "math.h"
#include "surf.h"
#include "dpm.h"
#include "stdio.h"

#define h -2.665 /* where curvature starts */

DEFINE_INIT(hook_setup,domain)
 {
    if (NULLP(user_particle_vars)) Init_User_Particle_Vars();
```

```
    /* now set the name and label */
        strcpy(user_particle_vars[0].name,"hook_zone_flag");
    strcpy(user_particle_vars[0].label,"Hook_Zone_Flag");
        strcpy(user_particle_vars[1].name,"hook_zone_time");
    strcpy(user_particle_vars[1].label,"Hook_Zone_Time");
 }


DEFINE_DPM_SCALAR_UPDATE(hook_capture, c, t, initialize, p)
{
        double hook_depth, thook;
        double xp, yp, zp, nf, wf_ir, wf_or, cap_ind, a, b;
        FILE *hookcaptured_nf_p, *hookcaptured_nf_n,*hookcaptured_ir, *
            hookcaptured_or;  /* hookcaptured_nf_p means the bubbles captured on the
            narrow face whose coordinate is positive x and vice cersa */

        /* You need to set the following parameter: */
        cap_ind = 4.;         /* if captured by hook, assign cap_ind = 4 */
        nf = 1.15;
        wf_ir =  0.15;
        wf_or =  -0.15;
        hook_depth = 0.0056;  /* set as 0.006 m */
        thook = 0.44;         /* set free travel time to be 0.25s */

        /* get partical (x,y,z) */
        xp = P_POS(p)[0];
        yp = P_POS(p)[1];
        zp = P_POS(p)[2];

        /* initialize the particle variables */
        /* if(initialize)
         {
                P_USER_REAL(p,0) = 0.;    Flag: particle entered hook zone? 0=No; 1=
                    Yes
                P_USER_REAL(p,1) = 0.;    Recording time enter the hook zone,
                    initialize to 0.
     } */


        /* Check if the particle is in the hook zone      */
        if(zp > 0 && yp > h)
        {
        if (nf - zp > hook_depth && xp - wf_or > hook_depth && wf_ir - xp >
            hook_depth)
        {
                P_USER_REAL(p,0) = 0; /* not in the hook zone */
        }
        else
        {
                /*if(P_VEL(p)[1] > 0)  ------comment: upgoing velocity? */
                /*{
                        NOTE: Half of mold region, only 3 walls with shell (+1
                            symmetric BC), so there are 3 if statement */
                        if(nf - zp < hook_depth)
                        {

                                if(P_USER_REAL(p,0) == 0)
                                {
```

```
                P_USER_REAL(p,0) = 1.;                    /*
                    OK, mark it as a particle in hook zone */
                P_USER_REAL(p,1) = P_TIME(p);   /* Time at
                    which particle entered hook zone */
        }
        if (P_USER_REAL(p,0) == 1. && P_TIME(p) -
            P_USER_REAL(p,1) >= thook) /* A particle already
            in hook zone for 0.25 sec? */
        {
                hookcaptured_nf_p = fopen("hook_nf_p.log","a
                    ");
                fprintf(hookcaptured_nf_p,"((%13.4e\t%13.4e\
                    t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                    %13.4e\t%13.4e\t%13.4e\t%13.4e\t%9.4f␣)
                    injection-0:%d)\n", P_POS(p)[0],P_POS(p)
                    [1],P_POS(p)[2],P_VEL(p)[0],P_VEL(p)[1],
                    P_VEL(p)[2],P_DIAM(p),0.,0.,cap_ind,P_T(p
                    ),P_TIME(p),p->part_id);
                p->stream_index = -1; /* tell FLUENT to give
                     up on this particle */
                MARK_PARTICLE(p, P_FL_REMOVED);
                fclose(hookcaptured_nf_p);
        }
}


if( xp - wf_or < hook_depth )
{
        if(P_USER_REAL(p,0) == 0)
        {
                P_USER_REAL(p,0) = 1.;                    /*
                    OK, mark it as a particle in hook zone */
                P_USER_REAL(p,1) = P_TIME(p);   /* Time at
                    which particle entered hook zone */
        }

        if (P_USER_REAL(p,0) == 1. && P_TIME(p) -
            P_USER_REAL(p,1) >= thook) /* A particle already
            in hook zone for 0.25 sec? */
        {
                hookcaptured_or = fopen("hook_or.log","a");
                fprintf(hookcaptured_or,"((%13.4e\t%13.4e\t
                    %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                    %13.4e\t%13.4e\t%13.4e\t%13.4e\t%9.4f␣)
                    injection-0:%d)\n", P_POS(p)[0],P_POS(p)
                    [1],P_POS(p)[2],P_VEL(p)[0],P_VEL(p)[1],
                    P_VEL(p)[2],P_DIAM(p),0.,0.,cap_ind,P_T(p
                    ),P_TIME(p),p->part_id);
                p->stream_index = -1; /* tell FLUENT to give
                     up on this particle */
                MARK_PARTICLE(p, P_FL_REMOVED);
                fclose(hookcaptured_or);
        }
}

if( wf_ir - xp < hook_depth)
{
        if(P_USER_REAL(p,0) == 0)
        {
```

```
                                P_USER_REAL(p,0) = 1.;                  /*
                                    OK, mark it as a particle in hook zone */
                                P_USER_REAL(p,1) = P_TIME(p);   /* record
                                    the time when it enter the zone */
                        }

                        if (P_USER_REAL(p,0) == 1. && P_TIME(p) -
                            P_USER_REAL(p,1) >= thook) /* A particle already
                            in hook zone for 0.25 sec? */
                        {
                                hookcaptured_ir = fopen("hook_ir.log","a");
                                fprintf(hookcaptured_ir,"((%13.4e\t%13.4e\t
                                    %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                    %13.4e\t%13.4e\t%13.4e\t%9.4f␣)
                                    injection-0:%d)\n", P_POS(p)[0],P_POS(p)
                                    [1],P_POS(p)[2],P_VEL(p)[0],P_VEL(p)[1],
                                    P_VEL(p)[2],P_DIAM(p),0.,0.,cap_ind,P_T(p
                                    ),P_TIME(p),p->part_id);
                                p->stream_index = -1; /* tell FLUENT to give
                                     up on this particle */
                                MARK_PARTICLE(p, P_FL_REMOVED);
                                fclose(hookcaptured_ir);
                        }
                }
}
}

if(zp < 0 && yp > h)
{
if (nf + zp > hook_depth && xp - wf_or > hook_depth && wf_ir - xp >
    hook_depth)
{
        P_USER_REAL(p,0) = 0; /* not in the hook zone */
}
else
{
        /*if(P_VEL(p)[1] > 0) ------comment: upgoing velocity? */
        /*{
                 NOTE: Half of mold region, only 3 walls with shell (+1
                     symmetric BC), so there are 3 if statement */
                if(nf + zp < hook_depth)
                {

                        if(P_USER_REAL(p,0) == 0)
                        {
                                P_USER_REAL(p,0) = 1.;                  /*
                                    OK, mark it as a particle in hook zone */
                                P_USER_REAL(p,1) = P_TIME(p);   /* Time at
                                    which particle entered hook zone */
                        }
                        if (P_USER_REAL(p,0) == 1. && P_TIME(p) -
                            P_USER_REAL(p,1) >= thook) /* A particle already
                            in hook zone for 0.25 sec? */
                        {
                                hookcaptured_nf_n = fopen("hook_nf_n.log","a
                                    ");
                                fprintf(hookcaptured_nf_n,"((%13.4e\t%13.4e\
                                    t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                    %13.4e\t%13.4e\t%13.4e\t%13.4e\t%9.4f␣)
```

```
                injection-0:%d)\n",P_POS(p)[0],P_POS(p)
                [1],P_POS(p)[2],P_VEL(p)[0],P_VEL(p)[1],
                P_VEL(p)[2],P_DIAM(p),0.,0.,cap_ind,P_T(p
                ),P_TIME(p),p->part_id);
            p->stream_index = -1; /* tell FLUENT to give
                 up on this particle */
            MARK_PARTICLE(p, P_FL_REMOVED);
            fclose(hookcaptured_nf_n);
        }
}


if( xp - wf_or < hook_depth )
{
        if(P_USER_REAL(p,0) == 0)
        {
            P_USER_REAL(p,0) = 1.;                    /*
                OK, mark it as a particle in hook zone */
            P_USER_REAL(p,1) = P_TIME(p);    /* Time at
                which particle entered hook zone */
        }

        if (P_USER_REAL(p,0) == 1. && P_TIME(p) -
            P_USER_REAL(p,1) >= thook) /* A particle already
            in hook zone for 0.25 sec? */
        {
            hookcaptured_or = fopen("hook_or.log","a");
            fprintf(hookcaptured_or,"((%13.4e\t%13.4e\t
                %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                %13.4e\t%13.4e\t%13.4e\t%9.4f_)
                injection-0:%d)\n", P_POS(p)[0],P_POS(p)
                [1],P_POS(p)[2],P_VEL(p)[0],P_VEL(p)[1],
                P_VEL(p)[2],P_DIAM(p),0.,0.,cap_ind,P_T(p
                ),P_TIME(p),p->part_id);
            p->stream_index = -1; /* tell FLUENT to give
                 up on this particle */
            MARK_PARTICLE(p, P_FL_REMOVED);
            fclose(hookcaptured_or);
        }
}

if( wf_ir - xp < hook_depth)
{
        if(P_USER_REAL(p,0) == 0)
        {
            P_USER_REAL(p,0) = 1.;                    /*
                OK, mark it as a particle in hook zone */
            P_USER_REAL(p,1) = P_TIME(p);    /* record
                the time when it enter the zone */
        }

        if (P_USER_REAL(p,0) == 1. && P_TIME(p) -
            P_USER_REAL(p,1) >= thook) /* A particle already
            in hook zone for 0.25 sec? */
        {
            hookcaptured_ir = fopen("hook_ir.log","a");
            fprintf(hookcaptured_ir,"((%13.4e\t%13.4e\t
                %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                %13.4e\t%13.4e\t%13.4e\t%13.4e\t%9.4f_)
```

```
                                        injection-0:%d)\n", P_POS(p)[0],P_POS(p)
                                        [1],P_POS(p)[2],P_VEL(p)[0],P_VEL(p)[1],
                                        P_VEL(p)[2],P_DIAM(p),0.,0.,cap_ind,P_T(p
                                        ),P_TIME(p),p->part_id);
                                    p->stream_index = -1; /* tell FLUENT to give
                                         up on this particle */
                                    MARK_PARTICLE(p, P_FL_REMOVED);
                                    fclose(hookcaptured_ir);
                                }
                            }
        }
        }
        }
```

Listing A.5: Entrapment / Engulfment Capture Mechanism and PDAS Calculation Used in Chapter 5

```
/*****************************************************************/
/* particle entrapment criteria UDF code for FLUENT  */
/* Checked and fixed bugs. Now using Random Walk if bubble is not captured*/
/* Final used version for after CCC 2013 */
/*****************************************************************/


/* Disable the warning c4996 from compiler using fscanf and fopen */
#pragma warning(disable : 4996)

#include "udf.h"
#include "mem.h"
#include "sg.h"
#include "math.h"
#include "surf.h"
#include "dpm.h"
#include "stdio.h"

double TM = 1533;  /* casting temperature in degree C */
double Ts = 1000;  /* interface temperature in degree C */
double k_steel = 26;     /* W/mK */
double rho_steel = 7000;  /* kg/m^3 */
double cp_steel = 680;    /* J/kgK */
double vc = 0.01;          /*casting speed in m/s*/
double PDAS_fitting_constant_wf = 0.000719671e-9;     /* in m^3c^2/s */
double PDAS_fitting_constant_nf = 0.000434549e-9;     /* in m^3c^2/s */

double Dragforce[3];
double Liftforce[3];
double AM_PG_SG[3];
/* double Liftforce[3]; */
FILE *pushlog;
double Cross_vel[3];
double Eta[3];
double Net_force_eta[3];
double Vel_diff_mag2;
double Drag_coeff;
double Drag_help;
double Rep;
int file_read = 1;
int bubbleid;
```

```
double cap_ind;
#define SIZE  176     /* size for v_sol interpolation, which contains the
    information for solidification front velocity */
#define SIZE_T_N 15  /* size for PDAS interpolation, PDAS distribution down the mold
     */

#define CurveNumber 8         /* Number of curvature parts in the curved strand */

double wide_shell_OR[CurveNumber+1] = {50, 10, 14, 17, 25, 28, 30, 33, 36};      /*
    Thread_ID for shell surface on WF_OR side, the elements are listed from menisucs
    to outlet */
double wide_shell_IR[CurveNumber+1] = {48, 9, 13, 16, 24, 27, 31, 34, 38};      /*
    Thread ID for shell surface on WF_IR side, the elements are listed from menisucs
    to outlet */
double narrow_shell[CurveNumber+1] = {52, 8, 12, 15, 19, 26, 32, 35, 44};       /*
    Thread ID for shell surface on NF side, the elements are listed from menisucs to
    outlet */

double x0[CurveNumber] = {84.85, 41.85, 27.55, 20.35, 16.15, 13.36, 11.36, 9.87};
           /* x xoordinate of the center of each curvature part */
double y_0[CurveNumber] = {-2.665, -2.799, -2.934, -3.071, -3.206, -3.342, -3.479,
    -3.619};         /* y xoordinate of the center of each curvature part */

double x_c_vsol[SIZE], y_c_vsol[SIZE], y_c_vsol_w[SIZE]; /* global declaration of
    solidification velocity vectors file narrow */
double x_c[SIZE], y_c[SIZE], y_c_w[SIZE];

/*****************************************************************/
/* UDF for specifying gas injection, gas sink at free surface, steel mass and
    momentum sink, */
/* and shear exchange and velocity continuity between steel and liquid slag layer */
/*****************************************************************/

#define casting_velocity 0.01
#define PI 3.14159

/*--------------------------------------------------------------------------------*/

/*function to read the solidification velocity on narrow face and wide face from a
    file*/
void reading_v_sol(double x_c[SIZE], double y_c[SIZE], double y_c_w[SIZE])
{
        FILE *fr;
        int i;
        double x_val, y_val, y_val_w;
        /* x_val = distance below meniscus */
        /* y_val = Vsol value on narrow face */
        /* y_val_w = Vsol value on wide face */

        char line[80];
        /* reading from the file v_sol_to_read.txt */
        fr = fopen ("v_sol_to_read.txt", "rt");
        i=0;
        while(fgets(line, 80, fr) != NULL)
        {
         sscanf (line, "%lf %lf %lf", &x_val, &y_val, &y_val_w);
         x_c[i] = x_val;
         y_c[i] = y_val;
         y_c_w[i] = y_val_w;
```

```
            i = i+1;
        }
    fclose(fr);
}

/*-------------------------------------------------------------------------*/
/*function to find the solidification velocity on narrow face and wide face by
    interpolation*/
double finding_vsol_inter(double y_pos, double x_c_v[SIZE], double y_c_v[SIZE])
{
        double v_sol;
        double x1, x2, y1, y2, x;
        int i;
        /* Doing linear interpolation by first finding between which two points
            y_pos exits */
        x = -(y_pos + 0.0);  /* to get the distance in term of distance below
            meniscus */
        for (i = 0; i < SIZE; i++)
        {
                if ( (x_c_v[i] < x) && (x < x_c_v[i+1]) )
                {
                        x1 = x_c_v[i];
                        x2 = x_c_v[i+1];
                        y1 = y_c_v[i];
                        y2 = y_c_v[i+1];

                        v_sol = ((x-x1)*y2 + (x2-x)*y1)/(x2-x1);
                        return (v_sol);
                }
        }

        v_sol = y_c_v[SIZE]; /* incase the particle hits way below in the mold  */
        return (v_sol);
}

/*
    ------------------------------------------------------------------------
    */

/*function to get the PDAS on the narrow face and wide face for a certian distance
    below meniscus*/
double finding_PDAS_inter(double y_pos, int PDAS_face, double Vsol)
{

        double PDAS;
        double x;
        double time;
        double dTdx;                                            /* Temperature
            gradient */
        double alpha_steel = k_steel / (rho_steel * cp_steel);   /* thermal
            diffusion */
        double k = 0.0032;          /* solidification coefficient in m/s^0.5 */
        double beta = k/(2*pow(alpha_steel,0.5));

        /* Doing linear interpolation by first finding between which two points
            y_pos exits */
        x = -(y_pos + 0.0);
        time = x/vc;
```

```
        dTdx = -(TM - Ts) / (exp(pow(beta, 2)) * erf(beta)* pow(PI*alpha_steel*time,
             0.5));   /* tempeartrue gradient in degree c/m */
        if (PDAS_face == 1)
        {
                PDAS = 4.3 * pow(PDAS_fitting_constant_nf, 0.25) * pow(Vsol, -0.25)
                    * pow(-dTdx, -0.5); /* narrow face wall */
        }
        else
        {
                PDAS = 4.3 * pow(PDAS_fitting_constant_wf, 0.25) * pow(Vsol, -0.25)
                    * pow(-dTdx, -0.5); /* wide face wall */
        }
        return (PDAS);
}

/*
    ------------------------------------------------------------------------------
    */

/*Macro that is used when the particle hits the nozzle walls to determine its
    position*/
DEFINE_DPM_BC(bc_nozzle_walls, p, t, f, f_normal, dim)
{
        FILE *fin;

        fin = fopen ("nozzle_boundary_hits.txt", "a");
        fprintf (fin, "P_POS(p)[0] is %e\t P_POS(p)[1] is %e\t P_POS(p)[2] %e\n" ,
            P_POS(p)[0], P_POS(p)[1], P_POS(p)[2]);
        fclose(fin);
        return (DPM_BC_TRAP);
}

/*Macro that is used when the particle hits the top surface to determine its
    postition*/
DEFINE_DPM_BC(bc_surface_top, p, t, f, f_normal, dim)
{
        FILE *fis;
        fis = fopen ("surface_boundary_hits.plt", "a");
        fprintf (fis, " %e\t %e\t %e\t %e\n" , P_POS(p)[0], P_POS(p)[1], P_POS(p)
            [2], P_DIAM(p));
        fclose(fis);
        return PATH_END;
}

DEFINE_DPM_BC(bc_outlet_hit, p, t, f, f_normal, dim)
{
        FILE *fis;
        fis = fopen ("outlet_boundary_hits.plt", "a");
        fprintf (fis, " %e\t %e\t %e\t %e\n" , P_POS(p)[0], P_POS(p)[1], P_POS(p)
            [2], P_DIAM(p));
        fclose(fis);
        return PATH_END;
}

/*Macro that is used everytime the particle hits the mold boundary walls to
    determine its fate*/
DEFINE_DPM_BC(bc_reflect_wf_ir, p, t, f, f_normal, dim)
{
        Thread *tff;
```

```
        cell_t c;
        Thread *tf;
        FILE *fi;
        FILE *fib;
        FILE *fdmp;
        FILE *fallforce;
        real x[ND_ND];
        int i, idim;
        double yoyo, yoyo2, yoyo3,Re,fe,Cd;
        int signyo2, signyo3;
        double Vsol, Rp, rd, F_lub, ho;
        double s_e, a_not, F_vand;
        double alpha, beta, zeta, n, Co, C_star, F_grad;
        double Ds, k, first_term, second_term, m;
        double B_W_force[3]; /* net buoyancy and weight force  */
        double theeta, F_tot_x[3], PDAS, Xi[3], Net_force_eta[3];
        double Rel_vel[3];
        double F_tot_x_try;
        double y_pos; /* y position in the mold where particle hits */
        double Velocity_diff[3];
        double Cross_vel2[3];
        int PDAS_face;

        double A[ND_ND];                                        /* area vector */
        double tauw, utau, yplus, uf, V_mag;
        double x1, y1, delta_x, delta_y, theta;
        int a;

        /* Calculate Lift force */
        double G, Gx, particle_dia, Reg, J, e_cons, L_star, L_w, Lift, signG, Lift_v
            [3];
        double Us, Rep;
        int signe;
        idim = dim;

        y_pos = P_POS(p)[1];
/*      printf("normal_vector%f%f",f_normal[0], f_normal[2]);*/
        if (file_read == 1)
        {
          reading_v_sol(x_c_vsol, y_c_vsol, y_c_vsol_w);                /* reading
              text file for solidification velocity */
          file_read = 2;                                                /* To read
              only once in the program*/
        }
        if ( fabs(f_normal[2]) > fabs(f_normal[0])) /* narrow face */
          {
            PDAS_face = 1;
            Vsol = finding_vsol_inter(y_pos, x_c_vsol, y_c_vsol);       /*
                interpolation for solidification velocity (narrow face) */
            PDAS = finding_PDAS_inter(y_pos, PDAS_face, Vsol);          /*
                Finds the PDAS value from a function (narrow face) */
          }
        else /* wide face*/
        {
            PDAS_face = -1;
            Vsol = finding_vsol_inter(y_pos, x_c_vsol, y_c_vsol_w);     /*
                interpolation for solidification velocity (wide face) */
            PDAS = finding_PDAS_inter(y_pos, PDAS_face, Vsol);          /*
                Finds the PDAS value from a function (wide face) */
```

```
  }
/* without even doing any force analysis, if particle diameter is smaller
    than PDAS, trap the particle if... */
if (P_DIAM(p) < PDAS)
{
        fdmp = fopen("wall-wfir.dpm","a");
        cap_ind = 0;
        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%9.4f )injection
            -0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],P_VEL(p)[0],P_VEL(p
            )[1],P_VEL(p)[2],P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,P_T(p),
            Dragforce[0], Dragforce[1], Dragforce[2], P_TIME(p),p->part_id);
        fclose(fdmp);
        return PATH_END;
}

particle_dia = P_DIAM(p);
c = P_CELL(p);
tff = P_CELL_THREAD(p);
Velocity_diff[0] = C_U(c,tff) - P_VEL(p)[0];
Velocity_diff[1] = C_V(c,tff) - P_VEL(p)[1];
Velocity_diff[2] = C_W(c,tff) - P_VEL(p)[2];
Rep=P_DIAM(p)*NV_MAG(Velocity_diff)*C_R(c,tff)/C_MU_L(c,tff);
Us = Rep * (C_MU_L(c,tff)/C_R(c,tff)) / particle_dia;

i=0;
G = C_DVDX(c,tff) + C_DWDX(c,tff);
if (G > 0)
{
        signG = 1;
}
else
{
        signG = -1;
}
Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff)); /* find the
    value of viscosity again */
e_cons = pow(Reg,0.5) / Rep;
if (e_cons<0)
{
        signe = -1;
}
else
{
        signe = 1;
}
if (0.1 < (signe*e_cons) < 20)
{
        J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
            e_cons-0.32)));
}
else /*else if (e_cons < 0.1) (signe*e_cons) <<1 */
{
        J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
}
Lift = (-9.0/PI)*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
    C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
Lift_v[0] = Lift;
```

```
Liftforce[0] = Lift;

i=1;
G = C_DUDY(c,tff) + C_DWDY(c,tff);
if (G > 0)
{
        signG = 1;
}
else
{
        signG = -1;
}
Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff)); /* find the
    value of viscosity again */
e_cons = pow(Reg,0.5) / Rep;
if (e_cons<0)
{
        signe = -1;
}
else
{
        signe = 1;
}
if (0.1 < (signe*e_cons) < 20)
{
        J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
            e_cons-0.32)));
}
else /* else if (e_cons < 0.1) (signe*e_cons) <<1 */
{
        J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
}
Lift = (-9.0/PI)*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
    C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
Lift_v[1] = Lift;
Liftforce[1] = Lift;

i=2;
G = C_DUDZ(c,tff) + C_DVDZ(c,tff);
if (G > 0)
{
        signG = 1;
}
else
{
        signG = -1;
}
Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff)); /* find the
    value of viscosity again */
e_cons = pow(Reg,0.5) / Rep;
if (e_cons<0)
{
        signe = -1;
}
else
{
        signe = 1;
}
if (0.1 < (signe*e_cons) < 20)
```

```
        {
                J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
                    e_cons-0.32)));
        }
        else /* else if (e_cons < 0.1) (signe*e_cons) <<1 */
        {
                J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
        }
        Lift = (-9.0/PI)*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
            C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
        Lift_v[2] = Lift;
        Liftforce[2] = Lift;


        /*lubrication force begins */
        Rp = P_DIAM(p)/2;          /* particle radius */
        rd = 0.0000033;            /* dendrite tip radius */
        /* ho is distance between dendrite tip and particle raduis .. it is assumed
            that this is much smaller than Rp and rd */
        ho = 7.84*pow(10,-8); /* for 400um ... ho = 6.22093*pow(10,-8); for 200um
            ... ho = 4.9e-8 for 100um particle according to Kaptay should be 4.9e-8;
            */
        F_lub = 6.0*PI*0.006*Vsol*(pow(Rp,2)/ho)*pow((rd/(Rp+rd)),2); /* lubrication
             force  */

        /*Interfacial force begins */
        s_e = 0.963; /* surface energy force */
        a_not = 2.5*pow(10,-10); /* atomic diameter of the liquid */
        F_vand = 2*PI*s_e*((rd*Rp)/(rd+Rp))*pow(a_not,2)/pow(ho,2); /* van derwall
            interfacial force */

        /* surface energy gradient force begins */
        n = 844;                   /* (1/mass%) */
        Co = 0.0028;      /* (mass%) */
        alpha = 1+ (n*Co);
        Ds = 3.4*pow(10,-9);       /* diffusion coefficient (m2/s) */
        k = 0.05;                                  /* Distribution coefficient (Cs/Cl)
            */
        C_star = Co / (1 - ((Vsol*rd)/(2*Ds))*(1-k));
        beta = n*rd*(C_star- Co);
        zeta = Rp + rd + ho;
        m = 0.171;

        first_term = -(m*beta*PI*Rp/pow(zeta,2)) * (  ((pow(zeta,2)-pow(Rp,2))/beta)
             * log( ((zeta+Rp)* (alpha*(zeta-Rp)+beta)) / ((zeta-Rp)* (alpha*(zeta+Rp
            )+beta)) ) );
        second_term = -(m*beta*PI*Rp/pow(zeta,2)) *  ( (2*Rp/alpha) - (beta/pow(
            alpha,2))*log( (alpha*(zeta+Rp)+beta) /  (alpha*(zeta-Rp)+beta) ) );
        F_grad = first_term + second_term; /*Surface energy gradient force*/

        /* net buoyancy and weight force begins */
        c = P_CELL(p);                    /* get the cell the particle is currently in
             */
    tff = P_CELL_THREAD(p); /* get the thread the particle is currently in */
        B_W_force[0] = 0.0;
/*      B_W_force[1] = ( C_R(c,t)- P_RHO(p)-0.5*C_R(c,t) ) * (4.0/3.0)* PI * pow(
    P_DIAM(p)/2,3) * 9.81;*/ /* upwards if the particle density is less than fluid
    density*/
```

```
    B_W_force[1] = ( C_R(c,tff)- P_RHO(p)  ) * (4.0/3.0)* PI * pow(P_DIAM(p)
        /2,3) * 9.81; /* upwards if the particle density is less than fluid
        density*/
    B_W_force[2] = 0.0;
    /* net buoyancy and weight force ends */

    /* Calculate Drag force */
    C_CENTROID(x,c,tff);
    Velocity_diff[0] = C_U(c,tff) - P_VEL(p)[0];
    Velocity_diff[1] = C_V(c,tff) - P_VEL(p)[1];
    Velocity_diff[2] = C_W(c,tff) - P_VEL(p)[2];


Re=P_DIAM(p)*NV_MAG(Velocity_diff)*C_R(c,tff)/C_MU_L(c,tff);
    fe = (1 + 0.15*pow(Re,0.687));  /* friction coefficient (Quan's Thesis)*/
    Cd = fe*(24/Re);                                  /* Drag coefficient (Quan's
        Thesis)   */
    Drag_help = (PI/8.0)*C_R(c,tff)*Cd*pow((Re*C_MU_L(c,tff)/C_R(c,tff)),2);
    Dragforce[0] = Drag_help * (Velocity_diff[0]) / NV_MAG(Velocity_diff);
    Dragforce[1] = Drag_help * (Velocity_diff[1]) / NV_MAG(Velocity_diff);
    Dragforce[2] = Drag_help * (Velocity_diff[2]) / NV_MAG(Velocity_diff);

    /*SETTING ESCAPE CRITERION*/
    c_face_loop(c,tff,i)
    {
      f = C_FACE(c,tff,i);
      tf = C_FACE_THREAD(c,tff,i);
      for (a = 0; a <= CurveNumber; a++) /* for-loop through all faces on WF_OR
          */
      {
      if(THREAD_ID(C_FACE_THREAD(c,tff,i)) == wide_shell_IR[a] )
            {
                    Xi[0] = -f_normal[0]; /* unit normal vector , face normal
                        vector*/
                    Xi[1] = -f_normal[1];
                    Xi[2] = -f_normal[2];
                    /*finding the Eta direction ( Sum of Bouyancy and Drag force
                         )*/

                    Cross_vel[0] = B_W_force[0] + Dragforce[0];
                    Cross_vel[1] = B_W_force[1] + Dragforce[1];
                    Cross_vel[2] = B_W_force[2] + Dragforce[2];
                    /*take dot product of Cross_vel with Xi .. then multiply
                        this number with Xi (unit vector) subtract this from
                        Cross_vel vector*/
                    /* to get Cross_vel2*/

                    Cross_vel2[0] = Cross_vel[0] - NV_DOT(Cross_vel,Xi)*Xi[0];
                    Cross_vel2[1] = Cross_vel[1] - NV_DOT(Cross_vel,Xi)*Xi[1];
                    Cross_vel2[2] = Cross_vel[2] - NV_DOT(Cross_vel,Xi)*Xi[2];

                    Eta[0] = Cross_vel2[0]/NV_MAG(Cross_vel2); /* getting unit
                        vector */
                    Eta[1] = Cross_vel2[1]/NV_MAG(Cross_vel2);
                    Eta[2] = Cross_vel2[2]/NV_MAG(Cross_vel2);
            }
      }
    }
```

```
theeta = asin(0.5*PDAS/(Rp+rd));
F_tot_x_try = NV_MAG(Liftforce) +  NV_DOT(B_W_force,Xi) + NV_DOT(Dragforce,
    Xi) - 2*(F_lub - F_grad - F_vand)*cos(theeta);
if (F_tot_x_try > 0.0)
{
        cap_ind = 4;
        fdmp = fopen("drift_back.dpm","a");
        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %9.4f )injection -0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],P_DIAM(p),0.,NV_MAG(Liftforce
            ),cap_ind,P_T(p),PDAS,NV_DOT(Dragforce,Xi),NV_DOT(B_W_force,Xi)
            ,2*(F_lub - F_grad - F_vand)*cos(theeta), P_TIME(p),p->part_id);
        fclose(fdmp);
        return PATH_ACTIVE;
}
else
{
        yoyo2 = NV_DOT(Dragforce,Eta);
        if (yoyo2 > 0)
        {
                signyo2 = 1;
        }
        else
        {
                signyo2 = -1;
        }

        yoyo3 = NV_DOT(B_W_force,Eta);
        if (yoyo3 > 0)
        {
                signyo3 = 1;
        }
        else
        {
                signyo3 = -1;
        }
        if (signyo2 == signyo3)
        {
                if      ((NV_DOT(Dragforce,Eta)*signyo2 + NV_DOT(B_W_force,
                    Eta)*signyo3)*cos(theeta) + (NV_MAG(Liftforce)+ NV_DOT(
                    Dragforce,Xi)+ NV_DOT(B_W_force,Xi))*sin(theeta) < (F_lub
                     - F_grad - F_vand)*sin(2*theeta))
                {
                        cap_ind = 1;
                        fdmp = fopen("wall-wfir.dpm","a");
                        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%9.4f )injection
                            -0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],P_DIAM(p),0.,
                            NV_MAG(Liftforce),cap_ind,P_T(p), Dragforce[0],
                            Dragforce[1], Dragforce[2], P_TIME(p),p->part_id)
                            ;
                        fclose(fdmp);
                        return PATH_END;
                }
                else
```

```
        {
                return PATH_ACTIVE;
        }
}
else
{
        if ( (signyo2*yoyo2) > (signyo3*yoyo3) )
        {
                if( (NV_DOT(Dragforce,Eta)*signyo2 - NV_DOT(
                    B_W_force,Eta)*signyo3)*cos(theeta) + (NV_MAG(
                    Liftforce)+ NV_DOT(Dragforce,Xi)+ NV_DOT(
                    B_W_force,Xi))*sin(theeta) < (F_lub - F_grad -
                    F_vand)*sin(2*theeta))
                {
                        cap_ind = 2;
                        fdmp = fopen("wall-wfir.dpm","a");
                        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%9.4f )injection-0:%d)\n"
                            ,P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],
                            P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,
                            P_T(p), Dragforce[0], Dragforce[1],
                            Dragforce[2], P_TIME(p),p->part_id);
                        fclose(fdmp);
                        return PATH_END;
                }
                else
                {
                        return PATH_ACTIVE;
                }
        }
        else
        {
                if ((NV_DOT(B_W_force,Eta)*signyo3 - NV_DOT(
                    Dragforce,Eta)*signyo2)*cos(theeta) + (NV_MAG(
                    Liftforce)+ NV_DOT(Dragforce,Xi)+ NV_DOT(
                    B_W_force,Xi))*sin(theeta) < (F_lub - F_grad -
                    F_vand)*sin(2*theeta))
                {
                        cap_ind = 3;
                        fdmp = fopen("wall-wfir.dpm","a");
                        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%9.4f )injection-0:%d)\n"
                            ,P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],
                            P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,
                            P_T(p), Dragforce[0], Dragforce[1],
                            Dragforce[2], P_TIME(p),p->part_id);
                        fclose(fdmp);
                        return PATH_END;
                }
                else
                {
```

```
                                        return PATH_ACTIVE;
                                    }
                                }
                            }
                        }
    return PATH_ACTIVE;
}

DEFINE_DPM_BC(bc_reflect_wf_or, p, t, f, f_normal, dim)
{
        /*face_t ff;*/
        Thread *tff;
        cell_t c;
        Thread *tf;
        FILE *fi;
        FILE *fib;
        FILE *fdmp;
        FILE *fallforce;
        real x[ND_ND];

        int i, idim;
        double yoyo, yoyo2, yoyo3,Re,fe,Cd;
        int signyo2, signyo3;
        double Vsol, Rp, rd, F_lub, ho;
        double s_e, a_not, F_vand;
        double alpha, beta, zeta, n, Co, C_star, F_grad;
        double Ds, k, first_term, second_term, m;
        double B_W_force[3];
        double theeta, F_tot_x[3], PDAS, Xi[3], Net_force_eta[3];
        double Rel_vel[3];
        double F_tot_x_try;
        double y_pos;
        double Velocity_diff[3];
        double Cross_vel2[3];
        int PDAS_face;

        double A[ND_ND];                                        /* area vector */
        double tauw, utau, yplus, uf, V_mag;
        double x1, y1, delta_x, delta_y, theta;
        int a;

        double G, particle_dia, Reg, J, e_cons, L_star, L_w, Lift, signG, Lift_v[3];
        double Us, Gx, Rep;
        int signe;
        idim = dim;
        y_pos = P_POS(p)[1];
        if (file_read == 1)
        {
          reading_v_sol(x_c_vsol, y_c_vsol, y_c_vsol_w);
          file_read = 2;
        }
        if ( fabs(f_normal[2]) > fabs(f_normal[0]))
          {
            PDAS_face = 1;
            Vsol = finding_vsol_inter(y_pos, x_c_vsol, y_c_vsol);
            PDAS = finding_PDAS_inter(y_pos, PDAS_face, Vsol);
          }
        else
          {
```

```
        PDAS_face = -1;
        Vsol = finding_vsol_inter(y_pos, x_c_vsol, y_c_vsol_w);
        PDAS = finding_PDAS_inter(y_pos, PDAS_face, Vsol);
      }

    if (P_DIAM(p) < PDAS)
    {
            cap_ind = 0;
            fdmp = fopen("wall-wfor.dpm","a");
            fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%9.4f )injection
                -0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],P_VEL(p)[0],P_VEL(p
                )[1],P_VEL(p)[2],P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,P_T(p),
                Dragforce[0], Dragforce[1], Dragforce[2], P_TIME(p),p->part_id);
            fclose(fdmp);
            return PATH_END;
    }

    particle_dia = P_DIAM(p);
c = P_CELL(p);
    tff = P_CELL_THREAD(p);
    Velocity_diff[0] = C_U(c,tff) - P_VEL(p)[0];
    Velocity_diff[1] = C_V(c,tff) - P_VEL(p)[1];
    Velocity_diff[2] = C_W(c,tff) - P_VEL(p)[2];
    Rep=P_DIAM(p)*NV_MAG(Velocity_diff)*C_R(c,tff)/C_MU_L(c,tff);
    Us = Rep * (C_MU_L(c,tff)/C_R(c,tff)) / particle_dia;


    i=0;
    G = C_DVDX(c,tff) + C_DWDX(c,tff);
    Gx = G;
    if (G > 0)
    {
            signG = 1;
    }
    else
    {
            signG = -1;
    }
    Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff));
    e_cons = pow(Reg,0.5) / Rep;
    if (e_cons<0)
    {
            signe = -1;
    }
    else
    {
            signe = 1;
    }
    if (0.1 < (signe*e_cons) < 20)
    {
            J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
                e_cons-0.32)));
    }
    else
    {
            J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
    }
```

```
Lift = (-9.0/PI)*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
    C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
Lift_v[0] = Lift;
Liftforce[0] = Lift;

i=1;
G = C_DUDY(c,tff) + C_DWDY(c,tff);
if (G > 0)
{
        signG = 1;
}
else
{
        signG = -1;
}
Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff));
e_cons = pow(Reg,0.5) / Rep;
if (e_cons<0)
{
        signe = -1;
}
else
{
        signe = 1;
}
if (0.1 < (signe*e_cons) < 20)
{
        J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
            e_cons-0.32)));
}
else
{
        J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
}
Lift = (-9.0/PI)*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
    C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
Lift_v[1] = Lift;
Liftforce[1] = Lift;

i=2;
G = C_DUDZ(c,tff) + C_DVDZ(c,tff);
if (G > 0)
{
        signG = 1;
}
else
{
        signG = -1;
}
Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff)); /* find the
    value of viscosity again*/
e_cons = pow(Reg,0.5) / Rep;
if (e_cons<0)
{
        signe = -1;
}
else
{
        signe = 1;
```

```
    }
    if (0.1 < (signe*e_cons) < 20)
    {
            J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
                e_cons-0.32)));
    }
    else
    {
            J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
    }
    Lift = (-9.0/PI)*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
        C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
    Lift_v[2] = Lift;
    Liftforce[2] = Lift;


    Rp = P_DIAM(p)/2;
    rd = 0.0000033;

    ho = 7.84*pow(10,-8);
    F_lub = 6.0*PI*0.006*Vsol*(pow(Rp,2)/ho)*pow((rd/(Rp+rd)),2);

    s_e = 0.963;
    a_not = 2.5*pow(10,-10);
    F_vand = 2*PI*s_e*((rd*Rp)/(rd+Rp))*pow(a_not,2)/pow(ho,2);


    n = 844;
    Co = 0.0028;
    alpha = 1+ (n*Co);
    Ds = 3.4*pow(10,-9);
    k = 0.05;
    C_star = Co / (1 - ((Vsol*rd)/(2*Ds))*(1-k));
    beta = n*rd*(C_star- Co);
    zeta = Rp + rd + ho;
    m = 0.171;
    first_term = -(m*beta*PI*Rp/pow(zeta,2)) * (  ((pow(zeta,2)-pow(Rp,2))/beta)
        * log( ((zeta+Rp)* (alpha*(zeta-Rp)+beta)) / ((zeta-Rp)* (alpha*(zeta+Rp
        )+beta)) ) );
    second_term = -(m*beta*PI*Rp/pow(zeta,2)) *  ( (2*Rp/alpha) - (beta/pow(
        alpha,2))*log( (alpha*(zeta+Rp)+beta) /  (alpha*(zeta-Rp)+beta) ) );
    F_grad = first_term + second_term;


    c = P_CELL(p);
tff = P_CELL_THREAD(p);
    B_W_force[0] = 0.0;

    B_W_force[1] = ( C_R(c,tff)- P_RHO(p)  ) * (4.0/3.0)* PI * pow(P_DIAM(p)
        /2,3) * 9.81;
    B_W_force[2] = 0.0;


    C_CENTROID(x,c,tff);
    Velocity_diff[0] = C_U(c,tff) - P_VEL(p)[0];
    Velocity_diff[1] = C_V(c,tff) - P_VEL(p)[1];
    Velocity_diff[2] = C_W(c,tff) - P_VEL(p)[2];

Re=P_DIAM(p)*NV_MAG(Velocity_diff)*C_R(c,tff)/C_MU_L(c,tff);
```

```
fe = (1 + 0.15*pow(Re,0.687));
Cd = fe*(24/Re);
Drag_help = (PI/8.0)*C_R(c,tff)*Cd*pow((Re*C_MU_L(c,tff)/C_R(c,tff)),2);
Dragforce[0] = Drag_help * (Velocity_diff[0]) / NV_MAG(Velocity_diff);
Dragforce[1] = Drag_help * (Velocity_diff[1]) / NV_MAG(Velocity_diff);
Dragforce[2] = Drag_help * (Velocity_diff[2]) / NV_MAG(Velocity_diff);

/*SETTING ESCAPE CRITERION*/
c_face_loop(c,tff,i)
{
  f = C_FACE(c,tff,i);
  tf = C_FACE_THREAD(c,tff,i);
  for (a = 0; a <= CurveNumber; a++) /* for-loop through all faces on WF_OR
      */
  {
  if(THREAD_ID(C_FACE_THREAD(c,tff,i)) == wide_shell_OR[a] )
        {
                Xi[0] = -f_normal[0];
                Xi[1] = -f_normal[1];
                Xi[2] = -f_normal[2];


                Cross_vel[0] = B_W_force[0] + Dragforce[0];
                Cross_vel[1] = B_W_force[1] + Dragforce[1];
                Cross_vel[2] = B_W_force[2] + Dragforce[2];


                Cross_vel2[0] = Cross_vel[0] - NV_DOT(Cross_vel,Xi)*Xi[0];
                Cross_vel2[1] = Cross_vel[1] - NV_DOT(Cross_vel,Xi)*Xi[1];
                Cross_vel2[2] = Cross_vel[2] - NV_DOT(Cross_vel,Xi)*Xi[2];

                Eta[0] = Cross_vel2[0]/NV_MAG(Cross_vel2);
                Eta[1] = Cross_vel2[1]/NV_MAG(Cross_vel2);
                Eta[2] = Cross_vel2[2]/NV_MAG(Cross_vel2);
        }
  }
}
theeta = asin(0.5*PDAS/(Rp+rd));
F_tot_x_try = NV_MAG(Liftforce) +  NV_DOT(B_W_force,Xi) + NV_DOT(Dragforce,
    Xi) - 2*(F_lub - F_grad - F_vand)*cos(theeta);
if (F_tot_x_try > 0.0)
{
        cap_ind = 4;
        fdmp = fopen("drift_back.dpm","a");
        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %9.4f )injection-0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],P_DIAM(p),0.,NV_MAG(Liftforce
            ),cap_ind,P_T(p),PDAS,NV_DOT(Dragforce,Xi),NV_DOT(B_W_force,Xi)
            ,2*(F_lub - F_grad - F_vand)*cos(theeta), P_TIME(p),p->part_id);
        fclose(fdmp);
        return PATH_ACTIVE;
}
else
{
        yoyo2 = NV_DOT(Dragforce,Eta);
        if (yoyo2 > 0)
        {
                signyo2 = 1;
```

```
}
else
{
        signyo2 = -1;
}
yoyo3 = NV_DOT(B_W_force,Eta);
if (yoyo3 > 0)
{
        signyo3 = 1;
}
else
{
        signyo3 = -1;
}
if (signyo2 == signyo3) /* both drageta and boyeta in the same
    direction*/
{
        if      ((NV_DOT(Dragforce,Eta)*signyo2 + NV_DOT(B_W_force,
            Eta)*signyo3)*cos(theeta) + (NV_MAG(Liftforce)+ NV_DOT(
            Dragforce,Xi)+ NV_DOT(B_W_force,Xi))*sin(theeta) < (F_lub
             - F_grad - F_vand)*sin(2*theeta))
        {
                cap_ind = 1;
                fdmp = fopen("wall-wfor.dpm","a");
                fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                    %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                    %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                    %13.4e\t%13.4e\t%13.4e\t%13.4e\t%9.4f_)injection
                    -0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                    P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],P_DIAM(p),0.,
                    NV_MAG(Liftforce),cap_ind,P_T(p),Dragforce[0],
                    Dragforce[1], Dragforce[2], P_TIME(p),p->part_id)
                    ;
                fclose(fdmp);
                return PATH_END;
        }
        else
        {
                return PATH_ACTIVE;
        }
}
else /* drageta and boyeta in opposite directions */
{
        if ( (signyo2*yoyo2) > (signyo3*yoyo3) )
        {
                if( (NV_DOT(Dragforce,Eta)*signyo2 - NV_DOT(
                    B_W_force,Eta)*signyo3)*cos(theeta) + (NV_MAG(
                    Liftforce)+ NV_DOT(Dragforce,Xi)+ NV_DOT(
                    B_W_force,Xi))*sin(theeta) < (F_lub - F_grad -
                    F_vand)*sin(2*theeta))
                {
                        /* fallforce = fopen("allforce.txt","a");*/
                        /* fprintf(fallforce,"Feta=%13.4e\tLift
                            =%13.4e\tRight=%13.4e",(NV_DOT(Dragforce,
                            Eta)*signyo2 - NV_DOT(B_W_force,Eta)*
                            signyo3)*cos(theeta),(NV_MAG(Liftforce)+
                            NV_DOT(Dragforce,Xi)+ NV_DOT(B_W_force,Xi
                            ))*sin(theeta),(F_lub - F_grad - F_vand)*
                            sin(2*theeta));*/
```

```
                                        /* fclose(fallforce); */
                                        cap_ind = 2;
                                        fdmp = fopen("wall-wfor.dpm","a");
                                        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%9.4f )injection-0:%d)\n"
                                            ,P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                                            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],
                                            P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,
                                            P_T(p), Dragforce[0], Dragforce[1],
                                            Dragforce[2], P_TIME(p),p->part_id);
                                        fclose(fdmp);
                                        return PATH_END;
                                }
                                else
                                {
                                        return PATH_ACTIVE;
                                }
                        }
                        else
                        {
                                if ((NV_DOT(B_W_force,Eta)*signyo3 - NV_DOT(
                                    Dragforce,Eta)*signyo2)*cos(theeta) + (NV_MAG(
                                    Liftforce)+ NV_DOT(Dragforce,Xi)+ NV_DOT(
                                    B_W_force,Xi))*sin(theeta) < (F_lub - F_grad -
                                    F_vand)*sin(2*theeta))
                                {
                                        cap_ind = 3;
                                        fdmp = fopen("wall-wfor.dpm","a");
                                        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%9.4f )injection-0:%d)\n"
                                            ,P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                                            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],
                                            P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,
                                            P_T(p),Dragforce[1], Dragforce[2], P_TIME
                                            (p),p->part_id);
                                        fclose(fdmp);
                                        return PATH_END;
                                }
                                else
                                {
                                        return PATH_ACTIVE;
                                }
                        }
                }
        }
    return PATH_ACTIVE;
}

DEFINE_DPM_BC(bc_reflect_nf, p, t, f, f_normal, dim)
{
        Thread *tff;
        cell_t c;
        Thread *tf;
```

```
FILE *fi;
FILE *fib;
FILE *fdmp;
FILE *fallforce;
real x[ND_ND];

int i, idim, iloop;
double yoyo, yoyo2, yoyo3,Re,fe,Cd;
int signyo2, signyo3;
double Vsol, Rp, rd, F_lub, ho;
double s_e, a_not, F_vand;
double alpha, beta, zeta, n, Co, C_star, F_grad;
double Ds, k, first_term, second_term, m;
double B_W_force[3];
double theeta, F_tot_x[3], PDAS, Xi[3], Net_force_eta[3];
double Rel_vel[3];
double F_tot_x_try;
double y_pos;
double Velocity_diff[3];
double Cross_vel2[3];
int PDAS_face;

double A[ND_ND];                                        /* area vector */
double tauw, utau, yplus, uf, V_mag;
double x1, y1, delta_x, delta_y, theta;
int a;


double G, particle_dia, Reg, J, e_cons, L_star, L_w, Lift, signG, Lift_v[3];
double Us, Gx, Rep;
int signe;
idim = dim;

printf ("reach_into_the_UDF_0");

y_pos = P_POS(p)[1];
if (file_read == 1)
{
  reading_v_sol(x_c_vsol, y_c_vsol, y_c_vsol_w);
  file_read = 2;
}

if ( fabs(f_normal[2]) > fabs(f_normal[0]))
  {
    PDAS_face = 1;
    Vsol = finding_vsol_inter(y_pos, x_c_vsol, y_c_vsol);
    PDAS = finding_PDAS_inter(y_pos, PDAS_face, Vsol);
  }
else
  {
    PDAS_face = -1;
    Vsol = finding_vsol_inter(y_pos, x_c_vsol, y_c_vsol_w);
    PDAS = finding_PDAS_inter(y_pos, PDAS_face, Vsol);
  }
if (P_DIAM(p) < PDAS)
{
        cap_ind = 0;
        fdmp = fopen("wall-nf.dpm","a");
```

```
            fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%9.4f␣)injection
                -0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],P_VEL(p)[0],P_VEL(p
                )[1],P_VEL(p)[2],P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,P_T(p),
                Dragforce[0], Dragforce[1], Dragforce[2], P_TIME(p),p->part_id);
            fclose(fdmp);
            return PATH_END;

    }


    particle_dia = P_DIAM(p);
c = P_CELL(p);
    tff = P_CELL_THREAD(p);
    Velocity_diff[0] = C_U(c,tff) - P_VEL(p)[0];
    Velocity_diff[1] = C_V(c,tff) - P_VEL(p)[1];
    Velocity_diff[2] = C_W(c,tff) - P_VEL(p)[2];
    Rep=P_DIAM(p)*NV_MAG(Velocity_diff)*C_R(c,tff)/C_MU_L(c,tff);
    Us = Rep * (C_MU_L(c,tff)/C_R(c,tff)) / particle_dia;

    i=0;
    G = C_DVDX(c,tff) + C_DWDX(c,tff);
    Gx = G;
    if (G > 0)
    {
            signG = 1;
    }
    else
    {
            signG = -1;
    }
    Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff));
    e_cons = pow(Reg,0.5) / Rep;
    if (e_cons<0)
    {
            signe = -1;
    }
    else
    {
            signe = 1;
    }
    if (0.1 < (signe*e_cons) < 20)
    {
            J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
                e_cons-0.32)));
    }
    else
    {
            J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
    }
    Lift = (-9.0/PI)*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
        C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
    Lift_v[0] = Lift;
    Liftforce[0] = Lift;

    i=1;
    G = C_DUDY(c,tff) + C_DWDY(c,tff);
    if (G > 0)
```

```
{
        signG = 1;
}
else
{
        signG = -1;
}
Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff));
e_cons = pow(Reg,0.5) / Rep;
if (e_cons<0)
{
        signe = -1;
}
else
{
        signe = 1;
}
if (0.1 < (signe*e_cons) < 20)
{
        J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
            e_cons-0.32)));
}
else
{
        J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
}
Lift = -9.0/PI*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
    C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
Lift_v[1] = Lift;
Liftforce[1] = Lift;

i=2;
G = C_DUDZ(c,tff) + C_DVDZ(c,tff);
if (G > 0)
{
        signG = 1;
}
else
{
        signG = -1;
}
Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff));
e_cons = pow(Reg,0.5) / Rep;
if (e_cons<0)
{
        signe = -1;
}
else
{
        signe = 1;
}
if (0.1 < (signe*e_cons) < 20)
{
        J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
            e_cons-0.32)));
}
else
{
        J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
```

```
    }
    Lift = -9.0/PI*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
        C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
    Lift_v[2] = Lift;
    Liftforce[2] = Lift;


    Rp = P_DIAM(p)/2;
    rd = 0.0000033;
    ho = 7.84*pow(10,-8);
    F_lub = 6.0*PI*0.006*Vsol*(pow(Rp,2)/ho)*pow((rd/(Rp+rd)),2);


    s_e = 0.963;
    a_not = 2.5*pow(10,-10);
    F_vand = 2*PI*s_e*((rd*Rp)/(rd+Rp))*pow(a_not,2)/pow(ho,2);

    n = 844;
    Co = 0.0028;
    alpha = 1+ (n*Co);
    Ds = 3.4*pow(10,-9);
    k = 0.05;
    C_star = Co / (1 - ((Vsol*rd)/(2*Ds))*(1-k));
    beta = n*rd*(C_star- Co);
    zeta = Rp + rd + ho;
    m = 0.171;
    first_term = -(m*beta*PI*Rp/pow(zeta,2)) * (  ((pow(zeta,2)-pow(Rp,2))/beta)
        * log( ((zeta+Rp)* (alpha*(zeta-Rp)+beta)) / ((zeta-Rp)* (alpha*(zeta+Rp
        )+beta)) ) );
    second_term = -(m*beta*PI*Rp/pow(zeta,2)) *  ( (2*Rp/alpha) - (beta/pow(
        alpha,2))*log( (alpha*(zeta+Rp)+beta) /  (alpha*(zeta-Rp)+beta) ) );
    F_grad = first_term + second_term;


    c = P_CELL(p);
tff = P_CELL_THREAD(p);
    B_W_force[0] = 0.0;

    B_W_force[1] = ( C_R(c,tff)- P_RHO(p)  ) * (4.0/3.0)* PI * pow(P_DIAM(p)
        /2,3) * 9.81;
    B_W_force[2] = 0.0;


    C_CENTROID(x,c,tff);

    Velocity_diff[0] = C_U(c,tff) - P_VEL(p)[0];
    Velocity_diff[1] = C_V(c,tff) - P_VEL(p)[1];
    Velocity_diff[2] = C_W(c,tff) - P_VEL(p)[2];

Re=P_DIAM(p)*NV_MAG(Velocity_diff)*C_R(c,tff)/C_MU_L(c,tff);
    fe = (1 + 0.15*pow(Re,0.687));
    Cd = fe*(24/Re);
    Drag_help = (PI/8.0)*C_R(c,tff)*Cd*pow((Re*C_MU_L(c,tff)/C_R(c,tff)),2);

    Dragforce[0] = Drag_help * (Velocity_diff[0]) / NV_MAG(Velocity_diff);
    Dragforce[1] = Drag_help * (Velocity_diff[1]) / NV_MAG(Velocity_diff);
    Dragforce[2] = Drag_help * (Velocity_diff[2]) / NV_MAG(Velocity_diff);
```

```
c_face_loop(c,tff,i)
{
  f = C_FACE(c,tff,i);
  tf = C_FACE_THREAD(c,tff,i);
  for (a = 0; a <= CurveNumber; a++) /* for-loop through all faces on WF_OR
      */
  {
  if(THREAD_ID(C_FACE_THREAD(c,tff,i)) == narrow_shell[a] )
        {
                Xi[0] = -f_normal[0];
                Xi[1] = -f_normal[1];
                Xi[2] = -f_normal[2];


                Cross_vel[0] = B_W_force[0] + Dragforce[0];
                Cross_vel[1] = B_W_force[1] + Dragforce[1];
                Cross_vel[2] = B_W_force[2] + Dragforce[2];


                Cross_vel2[0] = Cross_vel[0] - NV_DOT(Cross_vel,Xi)*Xi[0];
                Cross_vel2[1] = Cross_vel[1] - NV_DOT(Cross_vel,Xi)*Xi[1];
                Cross_vel2[2] = Cross_vel[2] - NV_DOT(Cross_vel,Xi)*Xi[2];

                Eta[0] = Cross_vel2[0]/NV_MAG(Cross_vel2);
                Eta[1] = Cross_vel2[1]/NV_MAG(Cross_vel2);
                Eta[2] = Cross_vel2[2]/NV_MAG(Cross_vel2);
        }
  }
}
theeta = asin(0.5*PDAS/(Rp+rd));
F_tot_x_try = NV_MAG(Liftforce) +  NV_DOT(B_W_force,Xi) + NV_DOT(Dragforce,
    Xi) - 2*(F_lub - F_grad - F_vand)*cos(theeta);
if (F_tot_x_try > 0.0)
{
        cap_ind = 4;
        fdmp = fopen("drift_back.dpm","a");
        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %9.4f_)injection-0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],P_DIAM(p),0.,NV_MAG(Liftforce
            ),cap_ind,P_T(p),PDAS,NV_DOT(Dragforce,Xi),NV_DOT(B_W_force,Xi)
             ,2*(F_lub - F_grad - F_vand)*cos(theeta), P_TIME(p),p->part_id);
        fclose(fdmp);
        return PATH_ACTIVE;
}
else
{
        yoyo2 = NV_DOT(Dragforce,Eta);
        if (yoyo2 > 0)
        {
                signyo2 = 1;
        }
        else
        {
                signyo2 = -1;
        }
        yoyo3 = NV_DOT(B_W_force,Eta);
        if (yoyo3 > 0)
        {
```

```
        signyo3 = 1;
}
else
{
        signyo3 = -1;
}
if (signyo2 == signyo3)
{
        if      ((NV_DOT(Dragforce,Eta)*signyo2 + NV_DOT(B_W_force,
            Eta)*signyo3)*cos(theeta) + (NV_MAG(Liftforce)+ NV_DOT(
            Dragforce,Xi)+ NV_DOT(B_W_force,Xi))*sin(theeta) < (F_lub
             - F_grad - F_vand)*sin(2*theeta))
        {
                cap_ind = 1;
                fdmp = fopen("wall-nf.dpm","a");
                fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                    %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                    %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                    %13.4e\t%13.4e\t%13.4e\t%13.4e\t%9.4f )injection
                    -0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                    P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],P_DIAM(p),0.,
                    NV_MAG(Liftforce),cap_ind,P_T(p), Dragforce[0],
                    Dragforce[1], Dragforce[2], P_TIME(p),p->part_id)
                    ;
                fclose(fdmp);
                return PATH_END;

        }
        else
        {
                return PATH_ACTIVE;
        }
}
else
{
        if ( (signyo2*yoyo2) > (signyo3*yoyo3) )
        {
                if( (NV_DOT(Dragforce,Eta)*signyo2 - NV_DOT(
                    B_W_force,Eta)*signyo3)*cos(theeta) + (NV_MAG(
                    Liftforce)+ NV_DOT(Dragforce,Xi)+ NV_DOT(
                    B_W_force,Xi))*sin(theeta) < (F_lub - F_grad -
                    F_vand)*sin(2*theeta))
                {
                        cap_ind = 2;
                        fdmp = fopen("wall-nf.dpm","a");
                        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%9.4f )injection -0:%d)\n"
                            ,P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],
                            P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,
                            P_T(p), Dragforce[0], Dragforce[1],
                            Dragforce[2], P_TIME(p),p->part_id);
                        fclose(fdmp);
                        return PATH_END;

                }
```

```
                                else
                                {
                                        return PATH_ACTIVE;
                                }
                        }
                        else
                        {
                                if ((NV_DOT(B_W_force,Eta)*signyo3 - NV_DOT(
                                    Dragforce,Eta)*signyo2)*cos(theeta) + (NV_MAG(
                                    Liftforce)+ NV_DOT(Dragforce,Xi)+ NV_DOT(
                                    B_W_force,Xi))*sin(theeta) < (F_lub - F_grad -
                                    F_vand)*sin(2*theeta))
                                {
                                        cap_ind = 3;
                                        fdmp = fopen("wall-nf.dpm","a");
                                        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%9.4f )injection-0:%d)\n"
                                            ,P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                                            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],
                                            P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,
                                            P_T(p),Dragforce[0], Dragforce[1],
                                            Dragforce[2], P_TIME(p),p->part_id);
                                        fclose(fdmp);
                                        return PATH_END;

                                }
                                else
                                {
                                        return PATH_ACTIVE;

                                }
                        }
                }
        }
    return PATH_ACTIVE;

}
/*
    /*-------------------------------------------------------------------------
     */

DEFINE_DPM_DRAG(particle_drag_force, Re, p) /* Macro for Drag force*/
{
        real w;
        double fe, Cd,  drag_force, Vel_diff[3], Vel_diff_mag, Us;

        double drag_check;
        double vf;

        FILE *f;

        f = fopen ("Mar12_drag.txt", "a");

        Rep = Re;

        {
```

```
        cell_t c = P_CELL(p);
        {
        Thread *t = P_CELL_THREAD(p);

        fe = (1 + 0.15*pow(Re,0.687));

        Cd = fe*(24/Re);



        drag_force = 18.0 * Cd * Re / 24.0 ;
        drag_check = (PI/8.0)*C_R(c,t)*Cd*pow((Re*C_MU_L(c,t)/C_R(c,t)),2);
        Drag_help = drag_check;
        Vel_diff_mag2 = (Re * C_MU_L(c,t)) / (P_DIAM(p) * C_R(c,t));
        Drag_coeff = Cd;
        vf = (Re*(C_MU_L(c,t)/C_R(c,t))/P_DIAM(p)); /* this is vf-vp */

        }
        }


        fclose(f);

        return(drag_force);

}


DEFINE_DPM_BODY_FORCE(DPMBF_Lift_added_mass, p, i)
{
/* /* Calculating Shear Lift Force */
FILE *fi;
double G, particle_dia, Reg, J, e, L_star, L_w, Lift, signG, Lift_v[3];
double Us, Gx;
int signe;
int ind;
double Velocity_diff[3];

ind = i;
Gx = 0;

cell_t c = P_CELL(p);
Thread *t = P_CELL_THREAD(p);
particle_dia = P_DIAM(p);
Velocity_diff[0] = C_U(c,t) - P_VEL(p)[0];
Velocity_diff[1] = C_V(c,t) - P_VEL(p)[1];
Velocity_diff[2] = C_W(c,t) - P_VEL(p)[2];
Rep=P_DIAM(p)*NV_MAG(Velocity_diff)*C_R(c,t)/C_MU_L(c,t);
Us = Rep * (C_MU_L(c,t)/C_R(c,t)) / particle_dia;

if (i == 0)
{
G = C_DVDX(c,t) + C_DWDX(c,t);
Gx = G;
}
else if (i == 1)
{
G = C_DUDY(c,t) + C_DWDY(c,t);
}
```

```
else
{
G = C_DUDZ(c,t) + C_DVDZ(c,t);
}

if (G > 0)
{
signG = 1;
}
else
{
signG = -1;
}

Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,t)/C_R(c,t)); /* find the value of
    viscosity again*/

e = pow(Reg,0.5) / Rep;

if (e<0)
{
        signe = -1;
}
else
{
        signe = 1;
}


if (0.1 < (signe*e) < 20)
{
J = 0.6765*(1+tanh( (2.5*log10(e)) + 0.191 ))*(0.667+ tanh(6*(e-0.32)));
}
else
{
J = -32.0*pow(PI,2)*pow(signe*e,5)*log(1/pow(e,2));
}

Lift = -9.0/PI*C_MU_L(c,t)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(C_MU_L(c,t)/
    C_R(c,t)),0.5)*J;

Lift_v[ind] = Lift;

Liftforce[ind] = Lift;

return (Lift/P_MASS(p));
}
```

Listing A.6: Entrapment / Engulfment Capture Mechanism and PDAS Calculation Used in Chapter 10

```
/*****************************************************************/
/* particle entrapment criteria UDF code for FLUENT  */
/* Checked and fixed bugs. Now using Random Walk if bubble is not captured*/
/* Final used version for after CCC 2013 */
/*****************************************************************/
```

```
/* Disable the warning c4996 from compiler using fscanf and fopen */
#pragma warning(disable : 4996)

#include "udf.h"
#include "mem.h"
#include "sg.h"
#include "math.h"
#include "surf.h"
#include "dpm.h"
#include "stdio.h"

double TM = 1533;  /* casting temperature in degree C */
double Ts = 1000;  /* interface temperature in degree C */
double k_steel = 26;     /* W/mK */
double rho_steel = 7000;  /* kg/m^3 */
double cp_steel = 680;    /* J/kgK */
double vc = 0.01;          /*casting speed in m/s*/
double PDAS_fitting_constant_wf = 0.000719671e-9;     /* in m^3c^2/s */
double PDAS_fitting_constant_nf = 0.000434549e-9;     /* in m^3c^2/s */

double kappa = 0.4187; /* von karman constant */
double EC = 9.793;  /* Empirical constant for log-law*/

double Dragforce[3];
double Liftforce[3];
double uf_v[3];
double up[3];
double AM_PG_SG[3];
/* double Liftforce[3]; */
FILE *pushlog;
double Cross_vel[3];
double Eta[3];
double Net_force_eta[3];
double Vel_diff_mag2;
double Drag_coeff;
double Drag_help;
double Rep;
int file_read = 1;
int bubbleid;
double cap_ind;
#define SIZE  176      /* size for v_sol interpolation, which contains the
    information for solidification front velocity */
#define SIZE_T_N 15  /* size for PDAS interpolation, PDAS distribution down the mold
     */

#define CurveNumber 8          /* Number of curvature parts in the curved strand */

double wide_shell_OR[CurveNumber+1] = {50, 10, 14, 17, 25, 28, 30, 33, 36};       /*
    Thread_ID for shell surface on WF_OR side, the elements are listed from menisucs
    to outlet */
double wide_shell_IR[CurveNumber+1] = {48, 9, 13, 16, 24, 27, 31, 34, 38};       /*
    Thread ID for shell surface on WF_IR side, the elements are listed from menisucs
    to outlet */
double narrow_shell[CurveNumber+1] = {52, 8, 12, 15, 19, 26, 32, 35, 44};       /*
    Thread ID for shell surface on NF side, the elements are listed from menisucs to
    outlet */

double x0[CurveNumber] = {84.85, 41.85, 27.55, 20.35, 16.15, 13.36, 11.36, 9.87};
           /* x xoordinate of the center of each curvature part */
```

```
double y_0[CurveNumber] = {-2.665, -2.799, -2.934, -3.071, -3.206, -3.342, -3.479,
    -3.619};          /* y xoordinate of the center of each curvature part */

double x_c_vsol[SIZE], y_c_vsol[SIZE], y_c_vsol_w[SIZE]; /* global declaration of
    solidification velocity vectors file narrow */
double x_c[SIZE], y_c[SIZE], y_c_w[SIZE];

/*****************************************************************/
/* UDF for specifying gas injection, gas sink at free surface, steel mass and
    momentum sink, */
/* and shear exchange and velocity continuity between steel and liquid slag layer */
/*****************************************************************/

#define casting_velocity 0.01
#define PI 3.14159

/*--------------------------------------------------------------------------*/

/*function to read the solidification velocity on narrow face and wide face from a
    file*/
void reading_v_sol(double x_c[SIZE], double y_c[SIZE], double y_c_w[SIZE])
{
        FILE *fr;
        int i;
        double x_val, y_val, y_val_w;
        /* x_val = distance below meniscus */
        /* y_val = Vsol value on narrow face */
        /* y_val_w = Vsol value on wide face */

        char line[80];
        /* reading from the file v_sol_to_read.txt */
        fr = fopen ("v_sol_to_read.txt", "rt");
        i=0;
        while(fgets(line, 80, fr) != NULL)
        {
         sscanf (line, "%lf %lf %lf", &x_val, &y_val, &y_val_w);
         x_c[i] = x_val;
         y_c[i] = y_val;
         y_c_w[i] = y_val_w;
         i = i+1;
        }
    fclose(fr);
}

/*--------------------------------------------------------------------------*/
/*function to find the solidification velocity on narrow face and wide face by
    interpolation*/
double finding_vsol_inter(double y_pos, double x_c_v[SIZE], double y_c_v[SIZE])
{
        double v_sol;
        double x1, x2, y1, y2, x;
        int i;
        /* Doing linear interpolation by first finding between which two points
            y_pos exits */
        x = -(y_pos + 0.0);  /* to get the distance in term of distance below
            meniscus */
        for (i = 0; i < SIZE; i++)
        {
                if ( (x_c_v[i] < x) && (x < x_c_v[i+1]) )
```

```
                {
                        x1 = x_c_v[i];
                        x2 = x_c_v[i+1];
                        y1 = y_c_v[i];
                        y2 = y_c_v[i+1];

                        v_sol = ((x-x1)*y2 + (x2-x)*y1)/(x2-x1);
                        return (v_sol);
                }
        }

        v_sol = y_c_v[SIZE]; /* incase the particle hits way below in the mold  */
        return (v_sol);
}

/*
    ------------------------------------------------------------------------
    */

/*function to get the PDAS on the narrow face and wide face for a certian distance
    below meniscus*/
double finding_PDAS_inter(double y_pos, int PDAS_face, double Vsol)
{

        double PDAS;
        double x;
        double time;
        double dTdx;                                                    /* Temperature
            gradient */
        double alpha_steel = k_steel / (rho_steel * cp_steel);    /* thermal
            diffusion */
        double k = 0.0032;          /* solidification coefficient in m/s^0.5 */
        double beta = k/(2*pow(alpha_steel,0.5));

        /* Doing linear interpolation by first finding between which two points
            y_pos exits */
        x = -(y_pos + 0.0);
        time = x/vc;
        dTdx = -(TM - Ts) / (exp(pow(beta, 2)) * erf(beta)* pow(PI*alpha_steel*time,
             0.5));    /* tempeartrue gradient in degree c/m */
        if (PDAS_face == 1)
        {
                PDAS = 4.3 * pow(PDAS_fitting_constant_nf, 0.25) * pow(Vsol, -0.25)
                    * pow(-dTdx, -0.5); /* narrow face wall */
        }
        else
        {
                PDAS = 4.3 * pow(PDAS_fitting_constant_wf, 0.25) * pow(Vsol, -0.25)
                    * pow(-dTdx, -0.5); /* wide face wall */
        }
        return (PDAS);
}

/*
    ------------------------------------------------------------------------
    */

/*Macro that is used when the particle hits the nozzle walls to determine its
    position*/
```

```
DEFINE_DPM_BC(bc_nozzle_walls, p, t, f, f_normal, dim)
{
        FILE *fin;

        fin = fopen ("nozzle_boundary_hits.txt", "a");
        fprintf (fin, "P_POS(p)[0] is %e\t P_POS(p)[1] is %e\t P_POS(p)[2] %e\n" ,
            P_POS(p)[0], P_POS(p)[1], P_POS(p)[2]);
        fclose(fin);
        return (DPM_BC_TRAP);
}

/*Macro that is used when the particle hits the top surface to determine its
    postition*/
DEFINE_DPM_BC(bc_surface_top, p, t, f, f_normal, dim)
{
        FILE *fis;
        fis = fopen ("surface_boundary_hits.plt", "a");
        fprintf (fis, " %e\t %e\t %e\t %e\n" , P_POS(p)[0], P_POS(p)[1], P_POS(p)
            [2], P_DIAM(p));
        fclose(fis);
        return PATH_END;
}

DEFINE_DPM_BC(bc_outlet_hit, p, t, f, f_normal, dim)
{
        FILE *fis;
        fis = fopen ("outlet_boundary_hits.plt", "a");
        fprintf (fis, " %e\t %e\t %e\t %e\n" , P_POS(p)[0], P_POS(p)[1], P_POS(p)
            [2], P_DIAM(p));
        fclose(fis);
        return PATH_END;
}

/*Macro that is used everytime the particle hits the mold boundary walls to
    determine its fate*/
DEFINE_DPM_BC(bc_reflect_wf_ir, p, t, f, f_normal, dim)
{
        Thread *tff;
        cell_t c;
        Thread *tf;
        FILE *fi;
        FILE *fib;
        FILE *fdmp;
        FILE *fallforce;
        real x[ND_ND];
        int i, idim;
        double yoyo, yoyo2, yoyo3,Re,fe,Cd;
        int signyo2, signyo3;
        double Vsol, Rp, rd, F_lub, ho;
        double s_e, a_not, F_vand;
        double alpha, beta, zeta, n, Co, C_star, F_grad;
        double Ds, k, first_term, second_term, m;
        double B_W_force[3]; /* net buoyancy and weight force  */
        double theeta, F_tot_x[3], PDAS, Xi[3], Net_force_eta[3];
        double Rel_vel[3];
        double F_tot_x_try;
        double y_pos; /* y position in the mold where particle hits */
        double Velocity_diff[3];
        double Cross_vel2[3];
```

```
int PDAS_face;

double A[ND_ND];                                        /* area vector */
double tauw, utau, yplus, uf, V_mag;
double x1, y1, delta_x, delta_y, theta;
int a;

/* Calculate Lift force */
double G, Gx, particle_dia, Reg, J, e_cons, L_star, L_w, Lift, signG, Lift_v
    [3];
double Us, Rep;
int signe;
idim = dim;

y_pos = P_POS(p)[1];
/*    printf("normal_vector%f%f",f_normal[0], f_normal[2]);*/
if (file_read == 1)
{
  reading_v_sol(x_c_vsol, y_c_vsol, y_c_vsol_w);                    /* reading
      text file for solidification velocity */
  file_read = 2;                                                    /* To read
      only once in the program*/
}
if ( fabs(f_normal[2]) > fabs(f_normal[0])) /* narrow face */
  {
    PDAS_face = 1;
    Vsol = finding_vsol_inter(y_pos, x_c_vsol, y_c_vsol);       /*
        interpolation for solidification velocity (narrow face) */
    PDAS = finding_PDAS_inter(y_pos, PDAS_face, Vsol);                /*
        Finds the PDAS value from a function (narrow face) */
  }
else /* wide face*/
{
    PDAS_face = -1;
    Vsol = finding_vsol_inter(y_pos, x_c_vsol, y_c_vsol_w);     /*
        interpolation for solidification velocity (wide face) */
    PDAS = finding_PDAS_inter(y_pos, PDAS_face, Vsol);                /*
        Finds the PDAS value from a function (wide face) */
  }
/* without even doing any force analysis, if particle diameter is smaller
    than PDAS, trap the particle if... */
if (P_DIAM(p) < PDAS)
{
        fdmp = fopen("wall-wfir.dpm","a");
        cap_ind = 0;
        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%9.4f )injection
            -0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],P_VEL(p)[0],P_VEL(p
            )[1],P_VEL(p)[2],P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,P_T(p),up
            [0], up[1], up[2], uf_v[0],uf_v[1],uf_v[2], Dragforce[0],
            Dragforce[1], Dragforce[2], P_TIME(p),p->part_id);
        fclose(fdmp);
        return PATH_END;
}

c = P_CELL(p);
tff = P_CELL_THREAD(p);
```

```
c_face_loop(c,tff,i)
{
  f = C_FACE(c,tff,i);
  tf = C_FACE_THREAD(c,tff,i);
  for (a = 0; a <= CurveNumber; a++) /* for-loop through all faces on WF_OR
      */
  {

  if(THREAD_ID(C_FACE_THREAD(c,tff,i)) == wide_shell_OR[a] )
        {
                /* calculate the particle velocity at solidification front
                   */
                if (a==0)
                {
                        up[0] = 0;      /*particle x velocity at
                            solidification front*/
                        up[1] = -vc;      /*particle y velocity at
                            solidification front*/
                        up[2] = 0;      /*particle z velocity at
                            solidification front*/
                }
                else
                {
                        F_CENTROID(x, f, tf);                   /*
                            F_CENTROID: generates coordinate position vector
                            x[ND_ND] as an output */
                        x1 = x[0];                          /* x coordinate
                            position of face centroid */
                        y1 = x[1];                          /* y coordinate
                            position of face centroid */
                        delta_x = fabs(x1-x0[a-1]);
                        delta_y = fabs(y1-y_0[a-1]);
                        theta = atan((delta_y)/(delta_x));              /*
                            the angle between the horizontal line (x
                            direction) and the line connecting the face-
                            center point and the acr-center point in radian
                            */
                        up[0] = vc * cos(PI/2 - theta);
                        up[1] = -vc * sin(PI/2 - theta);
                        up[2] = 0;

                }

                /* calcuate the fluid velocity at partcle center by strandrd
                     wall functoin */
                real NV_VEC(A);
                F_AREA(A, f, tf);
                                                                /*
                    F_AREA: generates area vector A[ND_ND] as an output */


                tauw=NV_MAG(F_STORAGE_R_N3V(f,tf, SV_WALL_SHEAR)) / NV_MAG(A
                    );      /* shear stress magnitude on the wall surface, and
                      it is assumed to be constant throughout the viscous
                    sublayer and log-law region*/
                utau = pow(tauw/C_R(c,tff),0.5);
                                                /* friction velocity
                    near the wall*/
```

```
                yplus = utau*P_DIAM(p)*C_R(c,tff)/2/C_MU_L(c,tff);
                                /* y+ value at the particle center */

                if (yplus < 11.225)   /* the particle is in viscous sub-
                    layer */
                {
                        uf = tauw * P_DIAM(p) / 2 /  C_MU_L(c,tff);    /*
                            fluid velocity magnitude at the particle center
                            */

                        V_mag = pow(C_U(c,tff),2) + pow(C_V(c,tff),2)+pow(
                            C_W(c,tff),2);
                        V_mag = pow(V_mag,0.5);   /* cell center fluid-
                            velocity magnitude */

                        uf_v[0] = uf * C_U(c,tff)/V_mag;     /* x-component
                            of fluid-velocity at the particle center */
                        uf_v[1] = uf * C_V(c,tff)/V_mag;     /* y-component
                            of fluid velocity at the particle center */
                        uf_v[2] = uf * C_W(c,tff)/V_mag;     /* z-component
                            of fluid velocity at the particle center */

                }
                else     /* the particle is in log-law region */
                {
                        uf = log(EC*utau*P_DIAM(p)*C_R(c,tff)/2/C_MU_L(c,tff
                            ));
                        uf = uf * utau / kappa;

                        V_mag = pow(C_U(c,tff),2) + pow(C_V(c,tff),2)+pow(
                            C_W(c,tff),2);
                        V_mag = pow(V_mag,0.5);   /* cell center fluid-
                            velocity magnitude */

                        uf_v[0] = uf * C_U(c,tff)/V_mag;     /* x-component
                            of fluid-velocity at the particle center */
                        uf_v[1] = uf * C_V(c,tff)/V_mag;     /* y-component
                            of fluid velocity at the particle center */
                        uf_v[2] = uf * C_W(c,tff)/V_mag;     /* z-component
                            of fluid velocity at the particle center */
                }

        }
  }
}

particle_dia = P_DIAM(p);

Velocity_diff[0] = uf_v[0] - up[0];
Velocity_diff[1] = uf_v[1] - up[1];
Velocity_diff[2] = uf_v[2] - up[2];
Rep=P_DIAM(p)*NV_MAG(Velocity_diff)*C_R(c,tff)/C_MU_L(c,tff);
Us = Rep * (C_MU_L(c,tff)/C_R(c,tff)) / particle_dia;

i=0;
G = C_DVDX(c,tff) + C_DWDX(c,tff);
if (G > 0)
{
        signG = 1;
```

```
}
else
{
        signG = -1;
}
Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff)); /* find the
    value of viscosity again */
e_cons = pow(Reg,0.5) / Rep;
if (e_cons<0)
{
        signe = -1;
}
else
{
        signe = 1;
}
if (0.1 < (signe*e_cons) < 20)
{
        J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
            e_cons-0.32)));
}
else /*else if (e_cons < 0.1) (signe*e_cons) <<1 */
{
        J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
}
Lift = (-9.0/PI)*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
    C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
Lift_v[0] = Lift;
Liftforce[0] = Lift;

i=1;
G = C_DUDY(c,tff) + C_DWDY(c,tff);
if (G > 0)
{
        signG = 1;
}
else
{
        signG = -1;
}
Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff)); /* find the
    value of viscosity again */
e_cons = pow(Reg,0.5) / Rep;
if (e_cons<0)
{
        signe = -1;
}
else
{
        signe = 1;
}
if (0.1 < (signe*e_cons) < 20)
{
        J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
            e_cons-0.32)));
}
else /* else if (e_cons < 0.1) (signe*e_cons) <<1 */
{
        J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
```

```
}
Lift = (-9.0/PI)*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
    C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
Lift_v[1] = Lift;
Liftforce[1] = Lift;

i=2;
G = C_DUDZ(c,tff) + C_DVDZ(c,tff);
if (G > 0)
{
        signG = 1;
}
else
{
        signG = -1;
}
Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff)); /* find the
    value of viscosity again */
e_cons = pow(Reg,0.5) / Rep;
if (e_cons<0)
{
        signe = -1;
}
else
{
        signe = 1;
}
if (0.1 < (signe*e_cons) < 20)
{
        J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
            e_cons-0.32)));
}
else /* else if (e_cons < 0.1) (signe*e_cons) <<1 */
{
        J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
}
Lift = (-9.0/PI)*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
    C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
Lift_v[2] = Lift;
Liftforce[2] = Lift;


/*lubrication force begins */
Rp = P_DIAM(p)/2;        /* particle radius */
rd = 0.0000033;          /* dendrite tip radius */
/* ho is distance between dendrite tip and particle raduis .. it is assumed
    that this is much smaller than Rp and rd */
ho = 7.84*pow(10,-8); /* for 400um ... ho = 6.22093*pow(10,-8); for 200um
    ... ho = 4.9e-8 for 100um particle according to Kaptay should be 4.9e-8;
    */
F_lub = 6.0*PI*0.006*Vsol*(pow(Rp,2)/ho)*pow((rd/(Rp+rd)),2); /* lubrication
     force  */

/*Interfacial force begins */
s_e = 0.963; /* surface energy force */
a_not = 2.5*pow(10,-10); /* atomic diameter of the liquid */
F_vand = 2*PI*s_e*((rd*Rp)/(rd+Rp))*pow(a_not,2)/pow(ho,2); /* van derwall
    interfacial force */
```

```
        /* surface energy gradient force begins */
        n = 844;                    /* (1/mass%) */
        Co = 0.0028;     /* (mass%) */
        alpha = 1+ (n*Co);
        Ds = 3.4*pow(10,-9);     /* diffusion coefficient (m2/s) */
        k = 0.05;                                   /* Distribution coefficient (Cs/Cl)
            */
        C_star = Co / (1 - ((Vsol*rd)/(2*Ds))*(1-k));
        beta = n*rd*(C_star- Co);
        zeta = Rp + rd + ho;
        m = 0.171;

        first_term = -(m*beta*PI*Rp/pow(zeta,2)) * (  ((pow(zeta,2)-pow(Rp,2))/beta)
             * log( ((zeta+Rp)* (alpha*(zeta-Rp)+beta)) / ((zeta-Rp)* (alpha*(zeta+Rp
            )+beta)) ) );
        second_term = -(m*beta*PI*Rp/pow(zeta,2)) *  ( (2*Rp/alpha) - (beta/pow(
            alpha,2))*log( (alpha*(zeta+Rp)+beta) /  (alpha*(zeta-Rp)+beta) ) );
        F_grad = first_term + second_term; /*Surface energy gradient force*/

        /* net buoyancy and weight force begins */
        c = P_CELL(p);                    /* get the cell the particle is currently in
             */
    tff = P_CELL_THREAD(p); /* get the thread the particle is currently in */
        B_W_force[0] = 0.0;
/*      B_W_force[1] = ( C_R(c,t)- P_RHO(p)-0.5*C_R(c,t) ) * (4.0/3.0)* PI * pow(
    P_DIAM(p)/2,3) * 9.81;*/ /* upwards if the particle density is less than fluid
    density*/
        B_W_force[1] = ( C_R(c,tff)- P_RHO(p)  ) * (4.0/3.0)* PI * pow(P_DIAM(p)
            /2,3) * 9.81; /* upwards if the particle density is less than fluid
            density*/
        B_W_force[2] = 0.0;
        /* net buoyancy and weight force ends */

        /* Calculate Drag force */
        C_CENTROID(x,c,tff);
        Velocity_diff[0] = uf_v[0] - up[0];
        Velocity_diff[1] = uf_v[1] - up[1];
        Velocity_diff[2] = uf_v[2] - up[2];

    Re=P_DIAM(p)*NV_MAG(Velocity_diff)*C_R(c,tff)/C_MU_L(c,tff);
        fe = (1 + 0.15*pow(Re,0.687));  /* friction coefficient (Quan's Thesis)*/
        Cd = fe*(24/Re);                                  /* Drag coefficient (Quan's
            Thesis)   */
        Drag_help = (PI/8.0)*C_R(c,tff)*Cd*pow((Re*C_MU_L(c,tff)/C_R(c,tff)),2);
        Dragforce[0] = Drag_help * (Velocity_diff[0]) / NV_MAG(Velocity_diff);
        Dragforce[1] = Drag_help * (Velocity_diff[1]) / NV_MAG(Velocity_diff);
        Dragforce[2] = Drag_help * (Velocity_diff[2]) / NV_MAG(Velocity_diff);

        /*SETTING ESCAPE CRITERION*/
        c_face_loop(c,tff,i)
        {
          f = C_FACE(c,tff,i);
          tf = C_FACE_THREAD(c,tff,i);
          for (a = 0; a <= CurveNumber; a++) /* for-loop through all faces on WF_OR
              */
          {
          if(THREAD_ID(C_FACE_THREAD(c,tff,i)) == wide_shell_IR[a] )
                {
```

```
                Xi[0] = -f_normal[0]; /* unit normal vector , face normal
                    vector*/
                Xi[1] = -f_normal[1];
                Xi[2] = -f_normal[2];
                /*finding the Eta direction ( Sum of Bouyancy and Drag force
                    )*/

                Cross_vel[0] = B_W_force[0] + Dragforce[0];
                Cross_vel[1] = B_W_force[1] + Dragforce[1];
                Cross_vel[2] = B_W_force[2] + Dragforce[2];
                /*take dot product of Cross_vel with Xi .. then multiply
                    this number with Xi (unit vector) subtract this from
                    Cross_vel vector*/
                /* to get Cross_vel2*/

                Cross_vel2[0] = Cross_vel[0] - NV_DOT(Cross_vel,Xi)*Xi[0];
                Cross_vel2[1] = Cross_vel[1] - NV_DOT(Cross_vel,Xi)*Xi[1];
                Cross_vel2[2] = Cross_vel[2] - NV_DOT(Cross_vel,Xi)*Xi[2];

                Eta[0] = Cross_vel2[0]/NV_MAG(Cross_vel2); /* getting unit
                    vector */
                Eta[1] = Cross_vel2[1]/NV_MAG(Cross_vel2);
                Eta[2] = Cross_vel2[2]/NV_MAG(Cross_vel2);
        }
  }
}

theeta = asin(0.5*PDAS/(Rp+rd));
F_tot_x_try = NV_MAG(Liftforce) +  NV_DOT(B_W_force,Xi) + NV_DOT(Dragforce,
    Xi) - 2*(F_lub - F_grad - F_vand)*cos(theeta);
if (F_tot_x_try > 0.0)
{
        cap_ind = 4;
        fdmp = fopen("drift_back.dpm","a");
        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %9.4f␣)injection-0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],P_DIAM(p),0.,NV_MAG(Liftforce
            ),cap_ind,P_T(p),PDAS,NV_DOT(Dragforce,Xi),NV_DOT(B_W_force,Xi)
            ,2*(F_lub - F_grad - F_vand)*cos(theeta), P_TIME(p),p->part_id);
        fclose(fdmp);
        return PATH_ACTIVE;
}
else
{
        yoyo2 = NV_DOT(Dragforce,Eta);
        if (yoyo2 > 0)
        {
                signyo2 = 1;
        }
        else
        {
                signyo2 = -1;
        }

        yoyo3 = NV_DOT(B_W_force,Eta);
        if (yoyo3 > 0)
        {
                signyo3 = 1;
```

```
}
else
{
        signyo3 = -1;
}
if (signyo2 == signyo3)
{
        if      ((NV_DOT(Dragforce,Eta)*signyo2 + NV_DOT(B_W_force,
            Eta)*signyo3)*cos(theeta) + (NV_MAG(Liftforce)+ NV_DOT(
            Dragforce,Xi)+ NV_DOT(B_W_force,Xi))*sin(theeta) < (F_lub
             - F_grad - F_vand)*sin(2*theeta))
        {
                cap_ind = 1;
                fdmp = fopen("wall-wfir.dpm","a");
                fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                    %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                    %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                    %13.4e\t%13.4e\t%13.4e\t%13.4e\t%9.4f␣)injection
                    -0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                    P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],P_DIAM(p),0.,
                    NV_MAG(Liftforce),cap_ind,P_T(p),up[0], up[1], up
                    [2], uf_v[0],uf_v[1],uf_v[2], Dragforce[0],
                    Dragforce[1], Dragforce[2], P_TIME(p),p->part_id)
                    ;
                fclose(fdmp);
                return PATH_END;
        }
        else
        {
                return PATH_ACTIVE;
        }
}
else
{
        if ( (signyo2*yoyo2) > (signyo3*yoyo3) )
        {
                if( (NV_DOT(Dragforce,Eta)*signyo2 - NV_DOT(
                    B_W_force,Eta)*signyo3)*cos(theeta) + (NV_MAG(
                    Liftforce)+ NV_DOT(Dragforce,Xi)+ NV_DOT(
                    B_W_force,Xi))*sin(theeta) < (F_lub - F_grad -
                    F_vand)*sin(2*theeta))
                {
                        cap_ind = 2;
                        fdmp = fopen("wall-wfir.dpm","a");
                        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%9.4f␣)injection-0:%d)\n"
                            ,P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],
                            P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,
                            P_T(p),up[0], up[1], up[2], uf_v[0],uf_v
                            [1],uf_v[2], Dragforce[0], Dragforce[1],
                            Dragforce[2], P_TIME(p),p->part_id);
                        fclose(fdmp);
                        return PATH_END;
                }
                else
```

```
                                {
                                        return PATH_ACTIVE;
                                }
                        }
                        else
                        {
                                if ((NV_DOT(B_W_force,Eta)*signyo3 - NV_DOT(
                                    Dragforce,Eta)*signyo2)*cos(theeta) + (NV_MAG(
                                    Liftforce)+ NV_DOT(Dragforce,Xi)+ NV_DOT(
                                    B_W_force,Xi))*sin(theeta) < (F_lub - F_grad -
                                    F_vand)*sin(2*theeta))
                                {
                                        cap_ind = 3;
                                        fdmp = fopen("wall-wfir.dpm","a");
                                        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%9.4f␣)injection-0:%d)\n"
                                            ,P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                                            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],
                                            P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,
                                            P_T(p),up[0], up[1], up[2], uf_v[0],uf_v
                                            [1],uf_v[2], Dragforce[0], Dragforce[1],
                                            Dragforce[2], P_TIME(p),p->part_id);
                                        fclose(fdmp);
                                        return PATH_END;
                                }
                                else
                                {
                                        return PATH_ACTIVE;
                                }
                        }
                }
        }
    return PATH_ACTIVE;
}

DEFINE_DPM_BC(bc_reflect_wf_or, p, t, f, f_normal, dim)
{
        /*face_t ff;*/
        Thread *tff;
        cell_t c;
        Thread *tf;
        FILE *fi;
        FILE *fib;
        FILE *fdmp;
        FILE *fallforce;
        real x[ND_ND];

        int i, idim;
        double yoyo, yoyo2, yoyo3,Re,fe,Cd;
        int signyo2, signyo3;
        double Vsol, Rp, rd, F_lub, ho;
        double s_e, a_not, F_vand;
        double alpha, beta, zeta, n, Co, C_star, F_grad;
        double Ds, k, first_term, second_term, m;
        double B_W_force[3];
        double theeta, F_tot_x[3], PDAS, Xi[3], Net_force_eta[3];
```

```
double Rel_vel[3];
double F_tot_x_try;
double y_pos;
double Velocity_diff[3];
double Cross_vel2[3];
int PDAS_face;

double A[ND_ND];                                        /* area vector */
double tauw, utau, yplus, uf, V_mag;
double x1, y1, delta_x, delta_y, theta;
int a;

double G, particle_dia, Reg, J, e_cons, L_star, L_w, Lift, signG, Lift_v[3];
double Us, Gx, Rep;
int signe;
idim = dim;
y_pos = P_POS(p)[1];
if (file_read == 1)
{
  reading_v_sol(x_c_vsol, y_c_vsol, y_c_vsol_w);
  file_read = 2;
}
if ( fabs(f_normal[2]) > fabs(f_normal[0]))
  {
    PDAS_face = 1;
    Vsol = finding_vsol_inter(y_pos, x_c_vsol, y_c_vsol);
    PDAS = finding_PDAS_inter(y_pos, PDAS_face, Vsol);
  }
else
  {
    PDAS_face = -1;
    Vsol = finding_vsol_inter(y_pos, x_c_vsol, y_c_vsol_w);
    PDAS = finding_PDAS_inter(y_pos, PDAS_face, Vsol);
  }

if (P_DIAM(p) < PDAS)
{
        cap_ind = 0;
        fdmp = fopen("wall-wfor.dpm","a");
        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%9.4f_)injection
            -0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],P_VEL(p)[0],P_VEL(p
            )[1],P_VEL(p)[2],P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,P_T(p),up
            [0], up[1], up[2], uf_v[0],uf_v[1],uf_v[2], Dragforce[0],
            Dragforce[1], Dragforce[2], P_TIME(p),p->part_id);
        fclose(fdmp);
        return PATH_END;
}

particle_dia = P_DIAM(p);
c = P_CELL(p);
tff = P_CELL_THREAD(p);

c_face_loop(c,tff,i)
{
  f = C_FACE(c,tff,i);
  tf = C_FACE_THREAD(c,tff,i);
```

```
for (a = 0; a <= CurveNumber; a++) /* for-loop through all faces on WF_OR
    */
{
if(THREAD_ID(C_FACE_THREAD(c,tff,i)) == wide_shell_OR[a] )
      {
              /* calculate the particle velocity at solidification front
                 */
              if (a==0)
              {
                      up[0] = 0;      /*particle x velocity at
                          solidification front*/
                      up[1] = -vc;      /*particle y velocity at
                          solidification front*/
                      up[2] = 0;      /*particle z velocity at
                          solidification front*/
              }
              else
              {
                      F_CENTROID(x, f, tf);                    /*
                          F_CENTROID: generates coordinate position vector
                          x[ND_ND] as an output */
                      x1 = x[0];                          /* x coordinate
                          position of face centroid */
                      y1 = x[1];                          /* y coordinate
                          position of face centroid */
                      delta_x = fabs(x1-x0[a-1]);
                      delta_y = fabs(y1-y_0[a-1]);
                      theta = atan((delta_y)/(delta_x));              /*
                          the angle between the horizontal line (x
                          direction) and the line connecting the face-
                          center point and the acr-center point in radian
                          */
                      up[0] = vc * cos(PI/2 - theta);
                      up[1] = -vc * sin(PI/2 - theta);
                      up[2] = 0;

              }

              /* calcuate the fluid velocity at partcle center by
                  strandard wall functoin */

              real NV_VEC(A);
              F_AREA(A, f, tf);
                                                              /*
                  F_AREA: generates area vector A[ND_ND] as an output */


              tauw=NV_MAG(F_STORAGE_R_N3V(f,tf, SV_WALL_SHEAR)) / NV_MAG(A
                  );     /* shear stress magnitude on the wall surface, and
                    it is assumed to be constant throughout the viscous
                  sublayer and log-law region*/
              utau = pow(tauw/C_R(c,tff),0.5);
                                                  /* friction velocity
                  near the wall*/
              yplus = utau*P_DIAM(p)*C_R(c,tff)/2/C_MU_L(c,tff);
                              /* y+ value at the particle center */

              if (yplus < 11.225)   /* the particle is in viscous sub-
                  layer */
```

```
                {
                        uf = tauw * P_DIAM(p) / 2 /  C_MU_L(c,tff);     /*
                            fluid velocity magnitude at the particle center
                            */

                        V_mag = pow(C_U(c,tff),2) + pow(C_V(c,tff),2)+pow(
                            C_W(c,tff),2);
                        V_mag = pow(V_mag,0.5);    /* cell center fluid-
                            velocity magnitude */

                        uf_v[0] = uf * C_U(c,tff)/V_mag;     /* x-component
                            of fluid-velocity at the particle center */
                        uf_v[1] = uf * C_V(c,tff)/V_mag;     /* y-component
                            of fluid velocity at the particle center */
                        uf_v[2] = uf * C_W(c,tff)/V_mag;     /* z-component
                            of fluid velocity at the particle center */

                }
                else     /* the particle is in log-law region */
                {
                        uf = log(EC*utau*P_DIAM(p)*C_R(c,tff)/2/C_MU_L(c,tff
                            ));
                        uf = uf * utau / kappa;

                        V_mag = pow(C_U(c,tff),2) + pow(C_V(c,tff),2)+pow(
                            C_W(c,tff),2);
                        V_mag = pow(V_mag,0.5);    /* cell center fluid-
                            velocity magnitude */

                        uf_v[0] = uf * C_U(c,tff)/V_mag;     /* x-component
                            of fluid-velocity at the particle center */
                        uf_v[1] = uf * C_V(c,tff)/V_mag;     /* y-component
                            of fluid velocity at the particle center */
                        uf_v[2] = uf * C_W(c,tff)/V_mag;     /* z-component
                            of fluid velocity at the particle center */
                }

        }
  }
}

Velocity_diff[0] = uf_v[0] - up[0];
Velocity_diff[1] = uf_v[1] - up[1];
Velocity_diff[2] = uf_v[2] - up[2];

Rep=P_DIAM(p)*NV_MAG(Velocity_diff)*C_R(c,tff)/C_MU_L(c,tff);
Us = Rep * (C_MU_L(c,tff)/C_R(c,tff)) / particle_dia;


i=0;
G = C_DVDX(c,tff) + C_DWDX(c,tff);
Gx = G;
if (G > 0)
{
        signG = 1;
}
else
{
        signG = -1;
```

```
}
Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff));
e_cons = pow(Reg,0.5) / Rep;
if (e_cons<0)
{
        signe = -1;
}
else
{
        signe = 1;
}
if (0.1 < (signe*e_cons) < 20)
{
        J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
            e_cons-0.32)));
}
else
{
        J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
}
Lift = (-9.0/PI)*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
    C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
Lift_v[0] = Lift;
Liftforce[0] = Lift;

i=1;
G = C_DUDY(c,tff) + C_DWDY(c,tff);
if (G > 0)
{
        signG = 1;
}
else
{
        signG = -1;
}
Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff));
e_cons = pow(Reg,0.5) / Rep;
if (e_cons<0)
{
        signe = -1;
}
else
{
        signe = 1;
}
if (0.1 < (signe*e_cons) < 20)
{
        J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
            e_cons-0.32)));
}
else
{
        J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
}
Lift = (-9.0/PI)*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
    C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
Lift_v[1] = Lift;
Liftforce[1] = Lift;
```

```
i=2;
G = C_DUDZ(c,tff) + C_DVDZ(c,tff);
if (G > 0)
{
        signG = 1;
}
else
{
        signG = -1;
}
Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff)); /* find the
    value of viscosity again*/
e_cons = pow(Reg,0.5) / Rep;
if (e_cons<0)
{
        signe = -1;
}
else
{
        signe = 1;
}
if (0.1 < (signe*e_cons) < 20)
{
        J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
            e_cons-0.32)));
}
else
{
        J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
}
Lift = (-9.0/PI)*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
    C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
Lift_v[2] = Lift;
Liftforce[2] = Lift;


Rp = P_DIAM(p)/2;
rd = 0.0000033;

ho = 7.84*pow(10,-8);
F_lub = 6.0*PI*0.006*Vsol*(pow(Rp,2)/ho)*pow((rd/(Rp+rd)),2);

s_e = 0.963;
a_not = 2.5*pow(10,-10);
F_vand = 2*PI*s_e*((rd*Rp)/(rd+Rp))*pow(a_not,2)/pow(ho,2);


n = 844;
Co = 0.0028;
alpha = 1+ (n*Co);
Ds = 3.4*pow(10,-9);
k = 0.05;
C_star = Co / (1 - ((Vsol*rd)/(2*Ds))*(1-k));
beta = n*rd*(C_star- Co);
zeta = Rp + rd + ho;
m = 0.171;
first_term = -(m*beta*PI*Rp/pow(zeta,2)) * (  ((pow(zeta,2)-pow(Rp,2))/beta)
     * log( ((zeta+Rp)* (alpha*(zeta-Rp)+beta)) / ((zeta-Rp)* (alpha*(zeta+Rp
    )+beta)) ) );
```

```
    second_term = -(m*beta*PI*Rp/pow(zeta,2)) *  ( (2*Rp/alpha) - (beta/pow(
        alpha,2))*log( (alpha*(zeta+Rp)+beta) /  (alpha*(zeta-Rp)+beta) ) );
    F_grad = first_term + second_term;


    c = P_CELL(p);
tff = P_CELL_THREAD(p);
    B_W_force[0] = 0.0;

    B_W_force[1] = ( C_R(c,tff)- P_RHO(p)  ) * (4.0/3.0)* PI * pow(P_DIAM(p)
        /2,3) * 9.81;
    B_W_force[2] = 0.0;


    C_CENTROID(x,c,tff);
    Velocity_diff[0] = uf_v[0] - up[0];
    Velocity_diff[1] = uf_v[1] - up[1];
    Velocity_diff[2] = uf_v[2] - up[2];

Re=P_DIAM(p)*NV_MAG(Velocity_diff)*C_R(c,tff)/C_MU_L(c,tff);
    fe = (1 + 0.15*pow(Re,0.687));
    Cd = fe*(24/Re);
    Drag_help = (PI/8.0)*C_R(c,tff)*Cd*pow((Re*C_MU_L(c,tff)/C_R(c,tff)),2);
    Dragforce[0] = Drag_help * (Velocity_diff[0]) / NV_MAG(Velocity_diff);
    Dragforce[1] = Drag_help * (Velocity_diff[1]) / NV_MAG(Velocity_diff);
    Dragforce[2] = Drag_help * (Velocity_diff[2]) / NV_MAG(Velocity_diff);

    /*SETTING ESCAPE CRITERION*/
    c_face_loop(c,tff,i)
    {
      f = C_FACE(c,tff,i);
      tf = C_FACE_THREAD(c,tff,i);
      for (a = 0; a <= CurveNumber; a++) /* for-loop through all faces on WF_OR
          */
      {
      if(THREAD_ID(C_FACE_THREAD(c,tff,i)) == wide_shell_OR[a] )
            {
                    Xi[0] = -f_normal[0];
                    Xi[1] = -f_normal[1];
                    Xi[2] = -f_normal[2];


                    Cross_vel[0] = B_W_force[0] + Dragforce[0];
                    Cross_vel[1] = B_W_force[1] + Dragforce[1];
                    Cross_vel[2] = B_W_force[2] + Dragforce[2];


                    Cross_vel2[0] = Cross_vel[0] - NV_DOT(Cross_vel,Xi)*Xi[0];
                    Cross_vel2[1] = Cross_vel[1] - NV_DOT(Cross_vel,Xi)*Xi[1];
                    Cross_vel2[2] = Cross_vel[2] - NV_DOT(Cross_vel,Xi)*Xi[2];

                    Eta[0] = Cross_vel2[0]/NV_MAG(Cross_vel2);
                    Eta[1] = Cross_vel2[1]/NV_MAG(Cross_vel2);
                    Eta[2] = Cross_vel2[2]/NV_MAG(Cross_vel2);
            }
      }
    }
    theeta = asin(0.5*PDAS/(Rp+rd));
```

```
F_tot_x_try = NV_MAG(Liftforce) + NV_DOT(B_W_force,Xi) + NV_DOT(Dragforce,
    Xi) - 2*(F_lub - F_grad - F_vand)*cos(theeta);
if (F_tot_x_try > 0.0)
{
        cap_ind = 4;
        fdmp = fopen("drift_back.dpm","a");
        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %9.4f_)injection-0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],P_DIAM(p),0.,NV_MAG(Liftforce
            ),cap_ind,P_T(p),PDAS,NV_DOT(Dragforce,Xi),NV_DOT(B_W_force,Xi)
            ,2*(F_lub - F_grad - F_vand)*cos(theeta), P_TIME(p),p->part_id);
        fclose(fdmp);
        return PATH_ACTIVE;
}
else
{
        yoyo2 = NV_DOT(Dragforce,Eta);
        if (yoyo2 > 0)
        {
                signyo2 = 1;
        }
        else
        {
                signyo2 = -1;
        }
        yoyo3 = NV_DOT(B_W_force,Eta);
        if (yoyo3 > 0)
        {
                signyo3 = 1;
        }
        else
        {
                signyo3 = -1;
        }
        if (signyo2 == signyo3) /* both drageta and boyeta in the same
            direction*/
        {
                if      ((NV_DOT(Dragforce,Eta)*signyo2 + NV_DOT(B_W_force,
                    Eta)*signyo3)*cos(theeta) + (NV_MAG(Liftforce)+ NV_DOT(
                    Dragforce,Xi)+ NV_DOT(B_W_force,Xi))*sin(theeta) < (F_lub
                     - F_grad - F_vand)*sin(2*theeta))
                {
                        cap_ind = 1;
                        fdmp = fopen("wall-wfor.dpm","a");
                        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%9.4f_)injection
                            -0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],P_DIAM(p),0.,
                            NV_MAG(Liftforce),cap_ind,P_T(p),up[0], up[1], up
                            [2], uf_v[0],uf_v[1],uf_v[2], Dragforce[0],
                            Dragforce[1], Dragforce[2], P_TIME(p),p->part_id)
                            ;
                        fclose(fdmp);
                        return PATH_END;
                }
                else
```

```
        {
                return PATH_ACTIVE;
        }
}
else /* drageta and boyeta in opposite directions */
{
        if ( (signyo2*yoyo2) > (signyo3*yoyo3) )
        {
                if( (NV_DOT(Dragforce,Eta)*signyo2 - NV_DOT(
                    B_W_force,Eta)*signyo3)*cos(theeta) + (NV_MAG(
                    Liftforce)+ NV_DOT(Dragforce,Xi)+ NV_DOT(
                    B_W_force,Xi))*sin(theeta) < (F_lub - F_grad -
                    F_vand)*sin(2*theeta))
                {
                        /* fallforce = fopen("allforce.txt","a");*/
                        /* fprintf(fallforce,"Feta=%13.4e\tLift
                            =%13.4e\tRight=%13.4e",(NV_DOT(Dragforce,
                            Eta)*signyo2 - NV_DOT(B_W_force,Eta)*
                            signyo3)*cos(theeta),(NV_MAG(Liftforce)+
                            NV_DOT(Dragforce,Xi)+ NV_DOT(B_W_force,Xi
                            ))*sin(theeta),(F_lub - F_grad - F_vand)*
                            sin(2*theeta));*/
                        /* fclose(fallforce); */
                        cap_ind = 2;
                        fdmp = fopen("wall-wfor.dpm","a");
                        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%9.4f )injection-0:%d)\n"
                            ,P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],
                            P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,
                            P_T(p),up[0], up[1], up[2], uf_v[0],uf_v
                            [1],uf_v[2], Dragforce[0], Dragforce[1],
                            Dragforce[2], P_TIME(p),p->part_id);
                        fclose(fdmp);
                        return PATH_END;
                }
                else
                {
                        return PATH_ACTIVE;
                }
        }
        else
        {
                if ((NV_DOT(B_W_force,Eta)*signyo3 - NV_DOT(
                    Dragforce,Eta)*signyo2)*cos(theeta) + (NV_MAG(
                    Liftforce)+ NV_DOT(Dragforce,Xi)+ NV_DOT(
                    B_W_force,Xi))*sin(theeta) < (F_lub - F_grad -
                    F_vand)*sin(2*theeta))
                {
                        cap_ind = 3;
                        fdmp = fopen("wall-wfor.dpm","a");
                        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%9.4f )injection-0:%d)\n"
```

```
                                        ,P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                                        P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],
                                        P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,
                                        P_T(p),up[0], up[1], up[2], uf_v[0],uf_v
                                        [1],uf_v[2], Dragforce[0], Dragforce[1],
                                        Dragforce[2], P_TIME(p),p->part_id);
                                fclose(fdmp);
                                return PATH_END;
                        }
                        else
                        {
                                return PATH_ACTIVE;
                        }
                }
        }
    }
    return PATH_ACTIVE;
}

DEFINE_DPM_BC(bc_reflect_nf, p, t, f, f_normal, dim)
{
        Thread *tff;
        cell_t c;
        Thread *tf;
        FILE *fi;
        FILE *fib;
        FILE *fdmp;
        FILE *fallforce;
        real x[ND_ND];

        int i, idim, iloop;
        double yoyo, yoyo2, yoyo3,Re,fe,Cd;
        int signyo2, signyo3;
        double Vsol, Rp, rd, F_lub, ho;
        double s_e, a_not, F_vand;
        double alpha, beta, zeta, n, Co, C_star, F_grad;
        double Ds, k, first_term, second_term, m;
        double B_W_force[3];
        double theeta, F_tot_x[3], PDAS, Xi[3], Net_force_eta[3];
        double Rel_vel[3];
        double F_tot_x_try;
        double y_pos;
        double Velocity_diff[3];
        double Cross_vel2[3];
        int PDAS_face;

        double A[ND_ND];                                        /* area vector */
        double tauw, utau, yplus, uf, V_mag;
        double x1, y1, delta_x, delta_y, theta;
        int a;


        double G, particle_dia, Reg, J, e_cons, L_star, L_w, Lift, signG, Lift_v[3];
        double Us, Gx, Rep;
        int signe;
        idim = dim;

        printf ("reach_into_the_UDF_0");
```

```
y_pos = P_POS(p)[1];
if (file_read == 1)
{
  reading_v_sol(x_c_vsol, y_c_vsol, y_c_vsol_w);
  file_read = 2;
}

if ( fabs(f_normal[2]) > fabs(f_normal[0]))
  {
    PDAS_face = 1;
    Vsol = finding_vsol_inter(y_pos, x_c_vsol, y_c_vsol);
   PDAS = finding_PDAS_inter(y_pos, PDAS_face, Vsol);
  }
else
  {
    PDAS_face = -1;
    Vsol = finding_vsol_inter(y_pos, x_c_vsol, y_c_vsol_w);
   PDAS = finding_PDAS_inter(y_pos, PDAS_face, Vsol);
  }
if (P_DIAM(p) < PDAS)
{
        cap_ind = 0;
        fdmp = fopen("wall-nf.dpm","a");
        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%9.4f )injection
            -0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],P_VEL(p)[0],P_VEL(p
            )[1],P_VEL(p)[2],P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,P_T(p),up
            [0], up[1], up[2], uf_v[0],uf_v[1],uf_v[2], Dragforce[0],
            Dragforce[1], Dragforce[2], P_TIME(p),p->part_id);
        fclose(fdmp);
        return PATH_END;

}


particle_dia = P_DIAM(p);
c = P_CELL(p);
tff = P_CELL_THREAD(p);

c_face_loop(c,tff,i)
{
  f = C_FACE(c,tff,i);
  tf = C_FACE_THREAD(c,tff,i);
  for (a = 0; a <= CurveNumber; a++) /* for-loop through all faces on WF_OR
      */
  {
  if(THREAD_ID(C_FACE_THREAD(c,tff,i)) == narrow_shell[a] )
        {
                /* calculate the particle velocity at solidification front
                    */
                if (a==0)
                {
                        up[0] = 0;      /*particle x velocity at
                            solidification front*/
                        up[1] = -vc;      /*particle y velocity at
                            solidification front*/
                        up[2] = 0;      /*particle z velocity at
                            solidification front*/
```

```
}
else
{
        F_CENTROID(x, f, tf);                    /*
            F_CENTROID: generates coordinate position vector
            x[ND_ND] as an output */
        x1 = x[0];                          /* x coordinate
            position of face centroid */
        y1 = x[1];                          /* y coordinate
            position of face centroid */
        delta_x = fabs(x1-x0[a-1]);
        delta_y = fabs(y1-y_0[a-1]);
        theta = atan((delta_y)/(delta_x));                /*
            the angle between the horizontal line (x
            direction) and the line connecting the face-
            center point and the acr-center point in radian
            */
        up[0] = vc * cos(PI/2 - theta);
        up[1] = -vc * sin(PI/2 - theta);
        up[2] = 0;

}

/* calcuate the fluid velocity at partcle center by
    strandard wall functoin */

real NV_VEC(A);
F_AREA(A, f, tf);
                                                    /*
    F_AREA: generates area vector A[ND_ND] as an output */


tauw=NV_MAG(F_STORAGE_R_N3V(f,tf, SV_WALL_SHEAR)) / NV_MAG(A
    );      /* shear stress magnitude on the wall surface, and
     it is assumed to be constant throughout the viscous
    sublayer and log-law region*/
utau = pow(tauw/C_R(c,tff),0.5);
                                        /* friction velocity
    near the wall*/
yplus = utau*P_DIAM(p)*C_R(c,tff)/2/C_MU_L(c,tff);
                    /* y+ value at the particle center */

if (yplus < 11.225)   /* the particle is in viscous sub-
    layer */
{
        uf = tauw * P_DIAM(p) / 2 /  C_MU_L(c,tff);     /*
            fluid velocity magnitude at the particle center
            */

        V_mag = pow(C_U(c,tff),2) + pow(C_V(c,tff),2)+pow(
            C_W(c,tff),2);
        V_mag = pow(V_mag,0.5);    /* cell center fluid-
            velocity magnitude */

        uf_v[0] = uf * C_U(c,tff)/V_mag;     /* x-component
            of fluid-velocity at the particle center */
        uf_v[1] = uf * C_V(c,tff)/V_mag;     /* y-component
            of fluid velocity at the particle center */
```

```
                        uf_v[2] = uf * C_W(c,tff)/V_mag;     /* z-component
                            of fluid velocity at the particle center */

                }
                else    /* the particle is in log-law region */
                {
                        uf = log(EC*utau*P_DIAM(p)*C_R(c,tff)/2/C_MU_L(c,tff
                            ));
                        uf = uf * utau / kappa;

                        V_mag = pow(C_U(c,tff),2) + pow(C_V(c,tff),2)+pow(
                            C_W(c,tff),2);
                        V_mag = pow(V_mag,0.5);   /* cell center fluid-
                            velocity magnitude */

                        uf_v[0] = uf * C_U(c,tff)/V_mag;     /* x-component
                            of fluid-velocity at the particle center */
                        uf_v[1] = uf * C_V(c,tff)/V_mag;     /* y-component
                            of fluid velocity at the particle center */
                        uf_v[2] = uf * C_W(c,tff)/V_mag;     /* z-component
                            of fluid velocity at the particle center */
                }

        }
  }
}

Velocity_diff[0] = uf_v[0] - up[0];
Velocity_diff[1] = uf_v[1] - up[1];
Velocity_diff[2] = uf_v[2] - up[2];

Rep=P_DIAM(p)*NV_MAG(Velocity_diff)*C_R(c,tff)/C_MU_L(c,tff);
Us = Rep * (C_MU_L(c,tff)/C_R(c,tff)) / particle_dia;

i=0;
G = C_DVDX(c,tff) + C_DWDX(c,tff);
Gx = G;
if (G > 0)
{
        signG = 1;
}
else
{
        signG = -1;
}
Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff));
e_cons = pow(Reg,0.5) / Rep;
if (e_cons<0)
{
        signe = -1;
}
else
{
        signe = 1;
}
if (0.1 < (signe*e_cons) < 20)
{
        J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
            e_cons-0.32)));
```

```
}
else
{
        J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
}
Lift = (-9.0/PI)*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
    C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
Lift_v[0] = Lift;
Liftforce[0] = Lift;

i=1;
G = C_DUDY(c,tff) + C_DWDY(c,tff);
if (G > 0)
{
        signG = 1;
}
else
{
        signG = -1;
}
Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff));
e_cons = pow(Reg,0.5) / Rep;
if (e_cons<0)
{
        signe = -1;
}
else
{
        signe = 1;
}
if (0.1 < (signe*e_cons) < 20)
{
        J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
            e_cons-0.32)));
}
else
{
        J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
}
Lift = -9.0/PI*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
    C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
Lift_v[1] = Lift;
Liftforce[1] = Lift;

i=2;
G = C_DUDZ(c,tff) + C_DVDZ(c,tff);
if (G > 0)
{
        signG = 1;
}
else
{
        signG = -1;
}
Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,tff)/C_R(c,tff));
e_cons = pow(Reg,0.5) / Rep;
if (e_cons<0)
{
        signe = -1;
```

```
    }
    else
    {
            signe = 1;
    }
    if (0.1 < (signe*e_cons) < 20)
    {
            J = 0.6765*(1+tanh( (2.5*log10(e_cons)) + 0.191 ))*(0.667 + tanh(6*(
                e_cons-0.32)));
    }
    else
    {
            J = -32.0*pow(PI,2)*pow(signe*e_cons,5)*log(1/pow(e_cons,2));
    }
    Lift = -9.0/PI*C_MU_L(c,tff)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(
        C_MU_L(c,tff)/C_R(c,tff)),0.5)*J;
    Lift_v[2] = Lift;
    Liftforce[2] = Lift;


    Rp = P_DIAM(p)/2;
    rd = 0.0000033;
    ho = 7.84*pow(10,-8);
    F_lub = 6.0*PI*0.006*Vsol*(pow(Rp,2)/ho)*pow((rd/(Rp+rd)),2);


    s_e = 0.963;
    a_not = 2.5*pow(10,-10);
    F_vand = 2*PI*s_e*((rd*Rp)/(rd+Rp))*pow(a_not,2)/pow(ho,2);

    n = 844;
    Co = 0.0028;
    alpha = 1+ (n*Co);
    Ds = 3.4*pow(10,-9);
    k = 0.05;
    C_star = Co / (1 - ((Vsol*rd)/(2*Ds))*(1-k));
    beta = n*rd*(C_star- Co);
    zeta = Rp + rd + ho;
    m = 0.171;
    first_term = -(m*beta*PI*Rp/pow(zeta,2)) * (  ((pow(zeta,2)-pow(Rp,2))/beta)
         * log( ((zeta+Rp)* (alpha*(zeta-Rp)+beta)) / ((zeta-Rp)* (alpha*(zeta+Rp
        )+beta)) ) );
    second_term = -(m*beta*PI*Rp/pow(zeta,2)) *  ( (2*Rp/alpha) - (beta/pow(
        alpha,2))*log( (alpha*(zeta+Rp)+beta) /  (alpha*(zeta-Rp)+beta) ) );
    F_grad = first_term + second_term;


    c = P_CELL(p);
tff = P_CELL_THREAD(p);
    B_W_force[0] = 0.0;

    B_W_force[1] = ( C_R(c,tff)- P_RHO(p)  ) * (4.0/3.0)* PI * pow(P_DIAM(p)
        /2,3) * 9.81;
    B_W_force[2] = 0.0;


    C_CENTROID(x,c,tff);

    Velocity_diff[0] = uf_v[0] - up[0];
```

```
    Velocity_diff[1] = uf_v[1] - up[1];
    Velocity_diff[2] = uf_v[2] - up[2];

Re=P_DIAM(p)*NV_MAG(Velocity_diff)*C_R(c,tff)/C_MU_L(c,tff);
    fe = (1 + 0.15*pow(Re,0.687));
    Cd = fe*(24/Re);
    Drag_help = (PI/8.0)*C_R(c,tff)*Cd*pow((Re*C_MU_L(c,tff)/C_R(c,tff)),2);

    Dragforce[0] = Drag_help * (Velocity_diff[0]) / NV_MAG(Velocity_diff);
    Dragforce[1] = Drag_help * (Velocity_diff[1]) / NV_MAG(Velocity_diff);
    Dragforce[2] = Drag_help * (Velocity_diff[2]) / NV_MAG(Velocity_diff);


    c_face_loop(c,tff,i)
    {
      f = C_FACE(c,tff,i);
      tf = C_FACE_THREAD(c,tff,i);
      for (a = 0; a <= CurveNumber; a++) /* for-loop through all faces on WF_OR
          */
      {
      if(THREAD_ID(C_FACE_THREAD(c,tff,i)) == narrow_shell[a] )
            {
                    Xi[0] = -f_normal[0];
                    Xi[1] = -f_normal[1];
                    Xi[2] = -f_normal[2];


                    Cross_vel[0] = B_W_force[0] + Dragforce[0];
                    Cross_vel[1] = B_W_force[1] + Dragforce[1];
                    Cross_vel[2] = B_W_force[2] + Dragforce[2];


                    Cross_vel2[0] = Cross_vel[0] - NV_DOT(Cross_vel,Xi)*Xi[0];
                    Cross_vel2[1] = Cross_vel[1] - NV_DOT(Cross_vel,Xi)*Xi[1];
                    Cross_vel2[2] = Cross_vel[2] - NV_DOT(Cross_vel,Xi)*Xi[2];

                    Eta[0] = Cross_vel2[0]/NV_MAG(Cross_vel2);
                    Eta[1] = Cross_vel2[1]/NV_MAG(Cross_vel2);
                    Eta[2] = Cross_vel2[2]/NV_MAG(Cross_vel2);
            }
      }
    }
    theeta = asin(0.5*PDAS/(Rp+rd));
    F_tot_x_try = NV_MAG(Liftforce) +  NV_DOT(B_W_force,Xi) + NV_DOT(Dragforce,
        Xi) - 2*(F_lub - F_grad - F_vand)*cos(theeta);
    if (F_tot_x_try > 0.0)
    {
            cap_ind = 4;
            fdmp = fopen("drift_back.dpm","a");
            fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                %9.4f_)injection-0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],P_DIAM(p),0.,NV_MAG(Liftforce
                ),cap_ind,P_T(p),PDAS,NV_DOT(Dragforce,Xi),NV_DOT(B_W_force,Xi)
                ,2*(F_lub - F_grad - F_vand)*cos(theeta), P_TIME(p),p->part_id);
            fclose(fdmp);
            return PATH_ACTIVE;
    }
    else
```

```
{
        yoyo2 = NV_DOT(Dragforce,Eta);
        if (yoyo2 > 0)
        {
                signyo2 = 1;
        }
        else
        {
                signyo2 = -1;
        }
        yoyo3 = NV_DOT(B_W_force,Eta);
        if (yoyo3 > 0)
        {
                signyo3 = 1;
        }
        else
        {
                signyo3 = -1;
        }
        if (signyo2 == signyo3)
        {
                if      ((NV_DOT(Dragforce,Eta)*signyo2 + NV_DOT(B_W_force,
                    Eta)*signyo3)*cos(theeta) + (NV_MAG(Liftforce)+ NV_DOT(
                    Dragforce,Xi)+ NV_DOT(B_W_force,Xi))*sin(theeta) < (F_lub
                     - F_grad - F_vand)*sin(2*theeta))
                {
                        cap_ind = 1;
                        fdmp = fopen("wall-nf.dpm","a");
                        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%9.4f )injection
                            -0:%d)\n",P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],P_DIAM(p),0.,
                            NV_MAG(Liftforce),cap_ind,P_T(p),up[0], up[1], up
                            [2], uf_v[0],uf_v[1],uf_v[2], Dragforce[0],
                            Dragforce[1], Dragforce[2], P_TIME(p),p->part_id)
                            ;
                        fclose(fdmp);
                        return PATH_END;

                }
                else
                {
                        return PATH_ACTIVE;
                }
        }
        else
        {
                if ( (signyo2*yoyo2) > (signyo3*yoyo3) )
                {
                        if( (NV_DOT(Dragforce,Eta)*signyo2 - NV_DOT(
                            B_W_force,Eta)*signyo3)*cos(theeta) + (NV_MAG(
                            Liftforce)+ NV_DOT(Dragforce,Xi)+ NV_DOT(
                            B_W_force,Xi))*sin(theeta) < (F_lub - F_grad -
                            F_vand)*sin(2*theeta))
                        {
                                cap_ind = 2;
                                fdmp = fopen("wall-nf.dpm","a");
```

```
                                        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%9.4f␣)injection=0:%d)\n"
                                            ,P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                                            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],
                                            P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,
                                            P_T(p),up[0], up[1], up[2], uf_v[0],uf_v
                                            [1],uf_v[2], Dragforce[0], Dragforce[1],
                                            Dragforce[2], P_TIME(p),p->part_id);
                                        fclose(fdmp);
                                        return PATH_END;

                                }
                                else
                                {
                                        return PATH_ACTIVE;
                                }
                        }
                        else
                        {
                                if ((NV_DOT(B_W_force,Eta)*signyo3 - NV_DOT(
                                    Dragforce,Eta)*signyo2)*cos(theeta) + (NV_MAG(
                                    Liftforce)+ NV_DOT(Dragforce,Xi)+ NV_DOT(
                                    B_W_force,Xi))*sin(theeta) < (F_lub - F_grad -
                                    F_vand)*sin(2*theeta))
                                {
                                        cap_ind = 3;
                                        fdmp = fopen("wall-nf.dpm","a");
                                        fprintf(fdmp,"((%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%13.4e\t%13.4e\t%13.4e\t
                                            %13.4e\t%13.4e\t%9.4f␣)injection=0:%d)\n"
                                            ,P_POS(p)[0],P_POS(p)[1],P_POS(p)[2],
                                            P_VEL(p)[0],P_VEL(p)[1],P_VEL(p)[2],
                                            P_DIAM(p),0.,NV_MAG(Liftforce),cap_ind,
                                            P_T(p),up[0], up[1], up[2], uf_v[0],uf_v
                                            [1],uf_v[2], Dragforce[0], Dragforce[1],
                                            Dragforce[2], P_TIME(p),p->part_id);
                                        fclose(fdmp);
                                        return PATH_END;

                                }
                                else
                                {
                                        return PATH_ACTIVE;

                                }
                        }
                }
        }
    return PATH_ACTIVE;

}
/*
    /*-------------------------------------------------------------------------
     */
```

```
DEFINE_DPM_DRAG(particle_drag_force, Re, p) /* Macro for Drag force*/
{
        real w;
        double fe, Cd,  drag_force, Vel_diff[3], Vel_diff_mag, Us;

        double drag_check;
        double vf;

        FILE *f;

        f = fopen ("Mar12_drag.txt", "a");

        Rep = Re;

        {
        cell_t c = P_CELL(p);
        {
        Thread *t = P_CELL_THREAD(p);

        fe = (1 + 0.15*pow(Re,0.687));

        Cd = fe*(24/Re);



        drag_force = 18.0 * Cd * Re / 24.0 ;
        drag_check = (PI/8.0)*C_R(c,t)*Cd*pow((Re*C_MU_L(c,t)/C_R(c,t)),2);
        Drag_help = drag_check;
        Vel_diff_mag2 = (Re * C_MU_L(c,t)) / (P_DIAM(p) * C_R(c,t));
        Drag_coeff = Cd;
        vf = (Re*(C_MU_L(c,t)/C_R(c,t))/P_DIAM(p)); /* this is vf-vp */

        }
        }


        fclose(f);

        return(drag_force);

}


DEFINE_DPM_BODY_FORCE(DPMBF_Lift_added_mass, p, i)
{
/* /* Calculating Shear Lift Force */
FILE *fi;
double G, particle_dia, Reg, J, e, L_star, L_w, Lift, signG, Lift_v[3];
double Us, Gx;
int signe;
int ind;
double Velocity_diff[3];

ind = i;
Gx = 0;

cell_t c = P_CELL(p);
Thread *t = P_CELL_THREAD(p);
```

```
particle_dia = P_DIAM(p);
Velocity_diff[0] = C_U(c,t) - P_VEL(p)[0];
Velocity_diff[1] = C_V(c,t) - P_VEL(p)[1];
Velocity_diff[2] = C_W(c,t) - P_VEL(p)[2];
Rep=P_DIAM(p)*NV_MAG(Velocity_diff)*C_R(c,t)/C_MU_L(c,t);
Us = Rep * (C_MU_L(c,t)/C_R(c,t)) / particle_dia;

if (i == 0)
{
G = C_DVDX(c,t) + C_DWDX(c,t);
Gx = G;
}
else if (i == 1)
{
G = C_DUDY(c,t) + C_DWDY(c,t);
}
else
{
G = C_DUDZ(c,t) + C_DVDZ(c,t);
}

if (G > 0)
{
signG = 1;
}
else
{
signG = -1;
}

Reg = signG*G*pow(particle_dia,2)/(C_MU_L(c,t)/C_R(c,t)); /* find the value of
    viscosity again*/

e = pow(Reg,0.5) / Rep;

if (e<0)
{
        signe = -1;
}
else
{
        signe = 1;
}


if (0.1 < (signe*e) < 20)
{
J = 0.6765*(1+tanh( (2.5*log10(e)) + 0.191 ))*(0.667+ tanh(6*(e-0.32)));
}
else
{
J = -32.0*pow(PI,2)*pow(signe*e,5)*log(1/pow(e,2));
}

Lift = -9.0/PI*C_MU_L(c,t)*pow(P_DIAM(p)/2,2)*Us*signG*pow((signG*G)/(C_MU_L(c,t)/
    C_R(c,t)),0.5)*J;

Lift_v[ind] = Lift;
```

```
Liftforce[ind] = Lift;

return (Lift/P_MASS(p));
}
```

The following appendix A.7 shows the solidification velocity (m/s) down the strand, on both NF and WF. First column: distance below the meniscus along the strand (m); Second column: solidification velocity on NF (m/s); Third column: solidification velocity on WF (m/s).

Listing A.7: Solidification Velocity

```
0       0.0011   0.0011
0.04    0.0008   0.0008
0.08    0.000565685      0.000565685
0.12    0.00046188       0.00046188
0.16    0.0004   0.0004
0.2     0.000357771      0.000357771
0.24    0.000326599      0.000326599
0.28    0.000302372      0.000302372
0.32    0.000282843      0.000282843
0.36    0.000266667      0.000266667
0.4     0.000252982      0.000252982
0.44    0.000241209      0.000241209
0.48    0.00023094       0.00023094
0.52    0.00022188       0.00022188
0.56    0.000213809      0.000213809
0.6     0.000206559      0.000206559
0.64    0.0002   0.0002
0.68    0.000194029      0.000194029
0.72    0.000188562      0.000188562
0.76    0.000183533      0.000183533
0.8     0.000178885      0.000178885
0.84    0.000174574      0.000174574
0.88    0.000170561      0.000170561
0.92    0.000166812      0.000166812
0.96    0.000163299      0.000163299
1       0.00016  0.00016
1.04    0.000156893      0.000156893
1.08    0.00015396       0.00015396
1.12    0.000151186      0.000151186
1.16    0.000148556      0.000148556
1.2     0.000146059      0.000146059
1.24    0.000143684      0.000143684
1.28    0.000141421      0.000141421
1.32    0.000139262      0.000139262
1.36    0.000137199      0.000137199
1.4     0.000135225      0.000135225
1.44    0.000133333      0.000133333
1.48    0.000131519      0.000131519
1.52    0.000129777      0.000129777
1.56    0.000128103      0.000128103
1.6     0.000126491      0.000126491
1.64    0.000124939      0.000124939
1.68    0.000123443      0.000123443
1.72    0.000121999      0.000121999
1.76    0.000120605      0.000120605
```

1.8	0.000119257	0.000119257
1.84	0.000117954	0.000117954
1.88	0.000116692	0.000116692
1.92	0.00011547	0.00011547
1.96	0.000114286	0.000114286
2	0.000113137	0.000113137
2.04	0.000112022	0.000112022
2.08	0.00011094	0.00011094
2.12	0.000109888	0.000109888
2.16	0.000108866	0.000108866
2.2	0.000107872	0.000107872
2.24	0.000106904	0.000106904
2.28	0.000105963	0.000105963
2.32	0.000105045	0.000105045
2.36	0.000104151	0.000104151
2.4	0.00010328	0.00010328
2.44	0.00010243	0.00010243
2.48	0.0001016	0.0001016
2.52	0.000100791	0.000100791
2.56	0.0001	0.0001
2.6	9.92278E−05	9.92278E−05
2.64	9.84732E−05	9.84732E−05
2.68	9.77356E−05	9.77356E−05
2.72	9.70143E−05	9.70143E−05
2.76	9.63087E−05	9.63087E−05
2.8	9.56183E−05	9.56183E−05
2.84	9.49425E−05	9.49425E−05
2.88	9.42809E−05	9.42809E−05
2.92	9.36329E−05	9.36329E−05
2.96	9.29981E−05	9.29981E−05
3	9.2376E−05	9.2376E−05
3.04	9.17663E−05	9.17663E−05
3.08	9.11685E−05	9.11685E−05
3.12	9.05822E−05	9.05822E−05
3.16	9.0007E−05	9.0007E−05
3.2	8.94427E−05	8.94427E−05
3.24	8.88889E−05	8.88889E−05
3.28	8.83452E−05	8.83452E−05
3.32	8.78114E−05	8.78114E−05
3.36	8.72872E−05	8.72872E−05
3.4	8.67722E−05	8.67722E−05
3.44	8.62662E−05	8.62662E−05
3.48	8.5769E−05	8.5769E−05
3.52	8.52803E−05	8.52803E−05
3.56	8.47998E−05	8.47998E−05
3.6	8.43274E−05	8.43274E−05
3.64	8.38628E−05	8.38628E−05
3.68	8.34058E−05	8.34058E−05
3.72	8.29561E−05	8.29561E−05
3.76	8.25137E−05	8.25137E−05
3.8	8.20783E−05	8.20783E−05
3.84	8.16497E−05	8.16497E−05
3.88	8.12277E−05	8.12277E−05
3.92	8.08122E−05	8.08122E−05
3.96	8.0403E−05	8.0403E−05
4	0.00008	0.00008
4.04	7.9603E−05	7.9603E−05
4.08	7.92118E−05	7.92118E−05
4.12	7.88263E−05	7.88263E−05

4.16	7.84465E–05	7.84465E–05
4.2	7.8072E–05	7.8072E–05
4.24	7.77029E–05	7.77029E–05
4.28	7.73389E–05	7.73389E–05
4.32	7.698E–05	7.698E–05
4.36	7.66261E–05	7.66261E–05
4.4	7.6277E–05	7.6277E–05
4.44	7.59326E–05	7.59326E–05
4.48	7.55929E–05	7.55929E–05
4.52	7.52577E–05	7.52577E–05
4.56	7.49269E–05	7.49269E–05
4.6	7.46004E–05	7.46004E–05
4.64	7.42781E–05	7.42781E–05
4.68	7.396E–05	7.396E–05
4.72	7.3646E–05	7.3646E–05
4.76	7.33359E–05	7.33359E–05
4.8	7.30297E–05	7.30297E–05
4.84	7.27273E–05	7.27273E–05
4.88	7.24286E–05	7.24286E–05
4.92	7.21336E–05	7.21336E–05
4.96	7.18421E–05	7.18421E–05
5	7.15542E–05	7.15542E–05
5.04	7.12697E–05	7.12697E–05
5.08	7.09885E–05	7.09885E–05
5.12	7.07107E–05	7.07107E–05
5.16	7.04361E–05	7.04361E–05
5.2	7.01646E–05	7.01646E–05
5.24	6.98963E–05	6.98963E–05
5.28	6.96311E–05	6.96311E–05
5.32	6.93688E–05	6.93688E–05
5.36	6.91095E–05	6.91095E–05
5.4	6.8853E–05	6.8853E–05
5.44	6.85994E–05	6.85994E–05
5.48	6.83486E–05	6.83486E–05
5.52	6.81005E–05	6.81005E–05
5.56	6.78551E–05	6.78551E–05
5.6	6.76123E–05	6.76123E–05
5.64	6.73722E–05	6.73722E–05
5.68	6.71345E–05	6.71345E–05
5.72	6.68994E–05	6.68994E–05
5.76	6.66667E–05	6.66667E–05
5.8	6.64364E–05	6.64364E–05
5.84	6.62085E–05	6.62085E–05
5.88	6.59829E–05	6.59829E–05
5.92	6.57596E–05	6.57596E–05
5.96	6.55386E–05	6.55386E–05
6	6.53197E–05	6.53197E–05
6.04	6.51031E–05	6.51031E–05
6.08	6.48886E–05	6.48886E–05
6.12	6.46762E–05	6.46762E–05
6.16	6.44658E–05	6.44658E–05
6.2	6.42575E–05	6.42575E–05
6.24	6.40513E–05	6.40513E–05
6.28	6.3847E–05	6.3847E–05
6.32	6.36446E–05	6.36446E–05
6.36	6.34441E–05	6.34441E–05
6.4	6.32456E–05	6.32456E–05
6.44	6.30488E–05	6.30488E–05
6.48	6.28539E–05	6.28539E–05

6.52	6.26608E–05	6.26608E–05
6.56	6.24695E–05	6.24695E–05
6.6	6.22799E–05	6.22799E–05
6.64	6.2092E–05	6.2092E–05
6.68	6.19059E–05	6.19059E–05
6.72	6.17213E–05	6.17213E–05
6.76	6.15385E–05	6.15385E–05
6.8	6.13572E–05	6.13572E–05
6.84	6.11775E–05	6.11775E–05
6.88	6.09994E–05	6.09994E–05
6.92	6.08229E–05	6.08229E–05
6.96	6.06478E–05	6.06478E–05
7	6.04743E–05	6.04743E–05

APPENDIX B

COPYRIGHT PERMISSIONS

B.1 Copyright Rules

Based on the AISTech website, permission is needed to reprint the papers published on AISTech proceedings.

The copyright agreement form the AISTech website, is shown in Figure B.1.

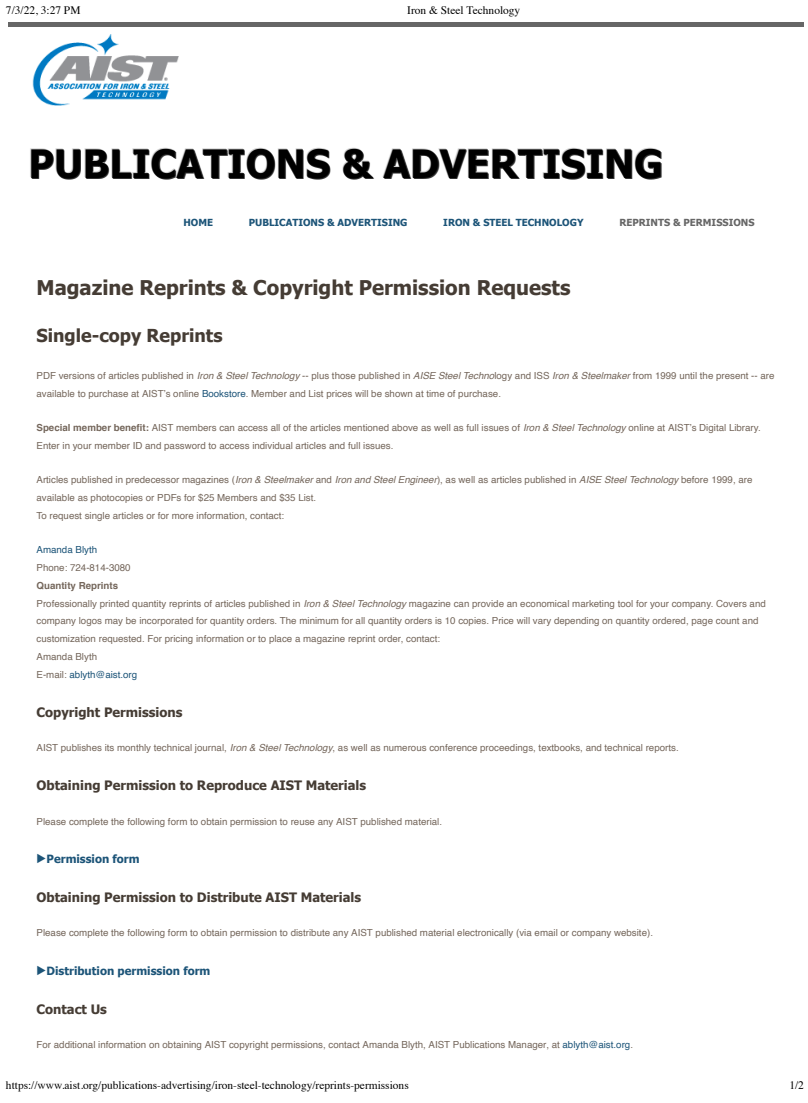


Figure B.1 Copy right permission stated on the AIST website

The copyright agreement form the MDPI website is shown in Figure B.2.

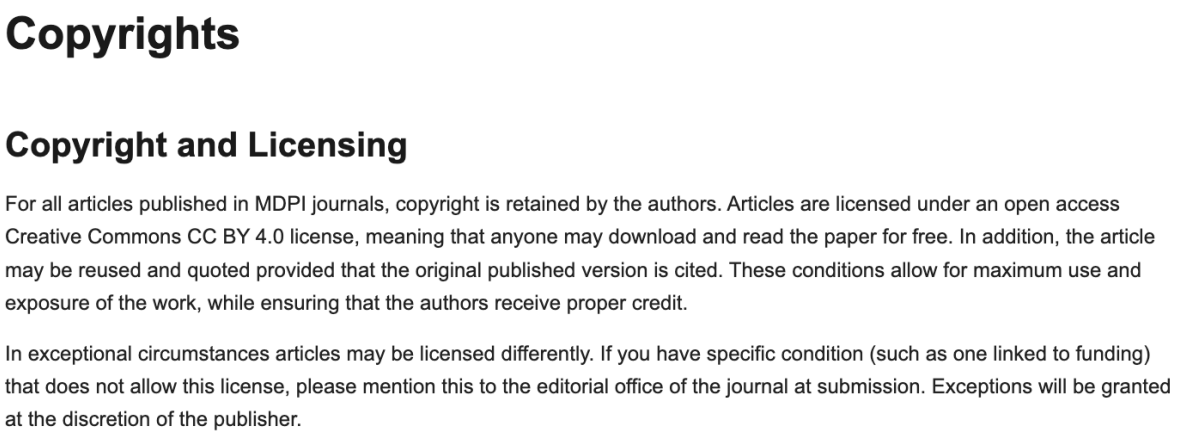


Figure B.2 Copy right permission stated on the MDPI website

B.2 AISTech 2019 Conference Paper

Here is where I present the permission from the conference AISTech and co-authors of the article entitled "Modeling of Multiphase Flow and Argon Bubble Entrapment in Continuous Slab Casting of Steel" published by the AISTech 2019 Proceedings, Volume 3, pages 2219 – 2231 (2019), which I include in the thesis under Chapter 3.

From: **Amanda Blyth** ablyth@aist.org
Subject: [External] RE: AISTech Paper Reprint Permisson for Mingyi Liang's Ph.D. Thesis
Date: June 29, 2022 at 5:01 AM
To: Mingyi Liang liang@mines.edu

AB

Hi Mingyi,

I'm sorry to have missed your call and that you have been waiting for this approval.

Yes, you have permission to use the paper in your thesis.

Kind regards,
Amanda

From: Mingyi Liang <liang@mines.edu>
Sent: Tuesday, June 28, 2022 5:22 PM
To: Amanda Blyth <ablyth@aist.org>
Subject: AISTech Paper Reprint Permisson for Mingyi Liang's Ph.D. Thesis

[External Email]

Hi Amanda,

Hope this email finds you well on top of your inbox.

I am a fifth-year Ph.D. candidate from Colorado School of Mines. Back in 2019, my co-workers and I published a paper on AISTech 2019 Proceedings, titled "Modeling of Multiphase Flow and Argon Bubble Entrapment in Continuous Slab Casting of Steel". As per the university's request, I will need to get the publisher's permission for the purpose of including the paper's results in my Ph.D. thesis, which should be done within the next a few days. I have submitted a permission form on AISTech website a few weeks ago but never heard back.

> Modeling of Multiphase Flow and Argon Bubble Entrapment in Continuous Slab Casting of Steel - Continuous Casting Consortium
>
> F(*d* Ü ; = *V* Ú*d* O*d* Ü ; *V* Ú L 1 - exp : F l *d* Ü *d* à Ø Ô á p á ; (1) where the mean diameter of the bubbles *d* à Ø Ô á is determined by a two-stage bubble size model [4] and the spread parameter, argon injected both into the UTN and upper plate for Case 1, and is 5.4 mm and 6.2 mm for the UTN and upper plate for
>
> ccc.illinois.edu

I am writing to check the progress and see if you could kindly grant me the permission to use this paper above and its results as a chapter in my Ph.D. thesis.

Thank you for your time and consideration.

Figure B.3 Copy right permission from the AIST publications manager, Amanda

BAOSTEEL 宝钢股份

BAOSHAN IRON & STEEL CO.LTD

Add: No.885 Administratitive Center,Fujin Road, Baoshan, Shanghai, China

Tel: +86 21 56784567

Fax: +86 21 26649000

Date: 6/21/2022

To whom it may concern,

I –Xiaoming Ruan as one of the co-authors of the article entitled "Modeling of Multiphase Flow and Argon Bubble Entrapment in Continuous Slab Casting of Steel" published in AISTech 2019 Proceedings, Volume 3, pages 2219 – 2231 (2019), permit Mr. Mingyi Liang to use the results of this paper as a chapter in his thesis.

Xiaoming Ruan

Figure B.4 Copy right permission from the co-author, Xiaoming Ruan

Date: June 30, 2022

To whom it may concern,

I, Seong-Mook Cho, as one of the co-authors of the article entitled "Modeling of Multiphase Flow and Argon Bubble Entrapment in Continuous Slab Casting of Steel" published in AISTech 2019 Proceedings, Volume 3, pages 2219 – 2231 (2019), permit Mr. Mingyi Liang to use the results of this paper as a chapter in his thesis.

Department of Metallurgical Engineering

Pukyong National University

Figure B.5 Copy right permission from the co-author, Seong-Mook Cho

COLORADOSCHOOLOFMINES
EARTH • ENERGY • ENVIRONMENT
Mechanical Engineering Department
1500 Illinois Street
Golden, Colorado 80401
mines.edu

Date: 6/28/2022

To whom it may concern,
I *Hamed Olia* as one of the co-authors of the article entitled "Modeling of Multiphase Flow and Argon Bubble Entrapment in Continuous Slab Casting of Steel" published in AISTech 2019 Proceedings, Volume 3, pages 2219 – 2231 (2019), permit Mr. Mingyi Liang to use the results of this paper as a chapter in his thesis.

Hamed Olia

Hamed Olia
Ph.D. & Research Assistant
Dept. of Mechanical Engineering
Colorado School of Mines
Brown Hall W470 I, 1610 Illinois Street, Golden, Colorado 80401
E-mail: holia@mines.edu

T: 303-273-3541
F: 303-273-3541

Figure B.6 Copy right permission from the co-author, Hamed Olia

Date: June, 24, 2022
To whom it may concern,

I, Lipsa Das, as one of the co-authors of the article entitled "Modeling of Multiphase Flow and Argon Bubble Entrapment in Continuous Slab Casting of Steel" published in AISTech 2019 Proceedings, Volume 3, pages 2219 – 2231 (2019), permit [redacted] Mr. Mingyi Liang to use the whole paper as a chapter in his thesis.

Lipsa Das

Department of Mechanical Engineering,
Colorado School of Mines

Figure B.7 Copy right permission from the co-author, Lipsa Das

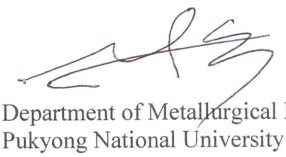
B.3 MDPI Journal (Processes) Paper

Here is where I present the permission from the co-authors of the article entitled "Modeling of Multiphase Flow, Superheat Dissipation, Particle Transport and Capture in a Vertical and Bending Continuous Caster" published by the MDPI journal (Processes) on July 21st, 2022, in the special issue "High-Efficiency and High-Quality Continuous Casting Processes – Special Issue", which I include in the thesis under Chapter 4. The permission from the MDPI journal, Processes, is not needed, because the author retains the copyright of this paper.

Date: July 04, 2022

To whom it may concern,

I, Seong-Mook Cho, as one of the co-authors of the article entitled "Modeling of Multiphase Flow, Superheat Dissipation, Particle Transport and Capture in a Vertical and Bending Continuous
Caster" submitted to the MDPI journal (Processes) on July 1st, 2022, in the special issue "High-Efficiency and High-Quality Continuous Casting Processes – Special Issue", permit Mr. Mingyi Liang to use this paper as a chapter in his thesis.



Department of Metallurgical Engineering
Pukyong National University

Figure B.8 Copy right permission from the co-author, Seong-Mook Cho

BAOSTEEL 宝钢股份

BAOSHAN IRON & STEEL CO.LTD

Add: No.885 Administratitive Center,Fujin Road, Baoshan, Shanghai, China

Tel: +86 21 56784567

Fax: +86 21 26649000

Date: 7/2/2022

To whom it may concern,

I - Xiaoming Ruan as one of the co-authors of the article entitled "Modeling of Multiphase Flow, Superheat Dissipation, Particle Transport and Capture in a Vertical and Bending Continuous Caster" submitted to the MDPI journal (Processes) on July 1st, 2022, in the special issue "High-Efficiency and High-Quality Continuous Casting Processes – Special Issue", permit Mr. Mingyi Liang to use this paper as a chapter in his thesis.

Figure B.9 Copy right permission from the co-author, Xiaoming Ruan

B.4 Permission for Figures

Below is the permission from Professor Brian G. Thomas who grants the author to reprint the figure 1.1 and 1.2 in this thesis.

Permission

Brian Thomas <bgthomas@mines.edu>
Mon 8/1/2022 3:00 PM

To: etdadministrator@proquest.com <etdadministrator@proquest.com>

Cc: Mingyi Liang <liang@mines.edu>

Dear Kristin Steinholt
Colorado School of Mines Administrator

I understand that you are inquiring regarding permission from my PhD student, Mingyi Liang, regarding Figs. 1.1 and 1.2 in his thesis.
He has my permission to use those figures in his thesis,
Best wishes,
--Brian.

Brian G. Thomas
Professor of Mechanical Engineering, Colorado School of Mines,
C.J. Gauthier Professor Emeritus and Research Professor, University of Illinois,
Department of Mechanical Engineering, Brown Hall W470-A
1610 Illinois Street, Golden, CO 80401
ph 303-273-3309; fax 303-273-3602
bgthomas@mines.edu; bgthomas@illinois.edu
http://ccc.illinois.edu (website)

Figure B.10 Figure reprint permission from professor Brian G. Thomas